VZ200-VZ300




VPROGRAMMEZ



VHINTZ



AND

VHARDWAREZ



NO.1



By John D'Alton

VPROGRAMMEZ

VHINTZ

AND

VHARDWAREZ

#1

PROGRAMME LISTINGS IN BASIC, ASSEMBLER AND MACHINE CODE.

HINTS AND HARDWARE FOR THE VZ200 AND VZ300 COLOUR COMPUTERS.

by John C.E.D'Alton.

Published by John D'Alton
39 Agnes St., TOOWONG. QLD. 4066. Australia.

Many of the items and articles were previously printed in the newsletter LE'VZ 200/300 DOP, and are reproduced in this book with the permission of the contributer.

CREDITS.

VZ200 and VZ300 are trademarks of Video Technology.
Z80 is a registered trademark of Zilog Inc.
Tandy and TRS80 are registered trademarks of the Tandy Corporation.
Microsoft is a registered trademark of Microsoft Inc.

I also give special thanks to contributors....

Mr.L.Taylor, Mr.A.Willows, Mr.R.Kitch, Mr.J.Perry, Mr.R.Small, Mr.P.Thursby, Mr.F.Olsen, Mr.C.Milner, Mr.G.Browell, Mr.G.Hall, Mr.H.Huggins.

I dedicate this book to my darling wife, Marie.

PREFACE

By purchasing this book you have shown more than a passing interest in computing. Perhaps you have grown tired of playing games on the VZ. With a certain amount of time taken to learn the BASIC language, you should be able to write your own games programmes. Of course there are many other practical uses that the VZ can be applied to. For this sort of information it is useful to join a users group (club) whereby you can talk direct to people with practical knowlege .

I have attempted to keep the programmes reasonably short, at least no longer than three pages. The first few are only a few lines long so that you can build up your typing skill and patience. The Machine Language (M/L) programmes or routines are for the advanced programmer, but there should be no reason why YOU should not be able to impliment those within a few months.

Then there are a few simple and not so simple hardware circuits for modifications or more advanced items.

In any case I hope YOU enjoy the contents of the book and perhaps introduce others to it.

John D'Alton.

CONTENTS.

NOTICE.

This is the third printing. June 1987.

The response from purchasers of this Book have been very favourable which of course is very pleasing to us. I have been asked by many when #2 will be published. If you would like me to publish another book, #2 would have different and more material, programmes, hardware, hints etc., please let me know.
In any case I have commenced gathering material for #2, but feedback from folk as to what they would like in it would be advantageous.
If you have anything to contribute then PLEASE sent it soon.

John D'Alton June 1987.

INTRODUCTION

Most of the BASIC programmes can be used with an unexpanded VZ200 (6K). The rest can be accommadated in an unexpanded VZ300 (18K)or an expanded VZ200 (22K).

I recommend the use of the special VZ Data Cassette Recorder which is especially designed to work with the VZ. There is no volume control to set and fiddle with, just play or record. Of course if you have the Disc Drive System the programmes are saved and loaded in a fraction of the time taken with the DTR.

It is a MUST that after you have typed in say a quarter of an hour of a programme to IMMEDIATLY <CSAVE> or <SAVE> if you have the Disc System BEFORE you LIST or RUN the programme (if you reached the end of it). ALWAYS save the partly typed programme with a name and a NUMBER. Say you start typing a programme called ADVENTURE. Call it "ADVENTURE 1". Then you can list or run it if you wish. Continue typing more of the programme and save it as "ADVENTURE 2", and so on until you have typed in the entire programme. Save it as ADVENTURE 7f", which means the seventh and final.

The reason for SAVING a typing session BEFORE listing and in particular RUNNING it, is that there may be (probably will be) mistakes in either your typing or the printing of the programme. If that is the case and you attemt to RUN the programme, the VZ may LOCK UP. That means that the VZ cannot carry out all the steps in it, and just can't continue, so there will be no flashing cursor or READY message. You will not be able to BREAK the VZ. You will not be able to SAVE what you have spent in the worst case hours to type in. If you did SAVE the programme, it's just a matter of switching off the VZ and loading back from the tape (or Disc) the programme and attempt to find the mistake or BUG.

Most programmes are for use with a tape based VZ, with the others suitable for a disc system.

You can modify some programmes to allow their use in your own programmes, in this way you will be learning programming at the same time. Some are badly written in an inefficient manner, so this also gives you more practice in tidying them up. Others are not games or complete programmes and are called routines. these can also be included in your own programmes.

There is other useful information such as communication addresses, PEEKs and POKES which will seem strange to a newcomer but are easy to use. There are twenty three Extended Basic Commands resident in the ROMs which can be implimented by POKES or by the use of the Ext. BASIC tape.

Warning!!! I will not take any responsibility for any damage caused by any hardware modification/s and/or addons. Any such hardware work is carried out at the owner/users risk.

TO START COMPUTING.

I do not intend teaching you all the basic operational and computing details which are discussed in the VZ200 and VZ300 Basic Reference Manuals (B.F.M.), but only to elaborate on some of the points that do seem to confuse the beginner. Always refer to the B.F.M. in conjuction with this book. I suggest that you start at the front of the B.F.M. and practice on the VZ until the end of the B.F.M. is reached.

There are some points that are not mentioned in the B.F.M. that are in this book that will make computing quite a lot easier. All programme listings are <LLISTings> directly from the programme, so the programme SHOULD be bug free. A BUG in a programme is an ERROR of FAULT.

EDITING.

One of the most important computing tasks that should be mastered very early is the EDITing function. This function on the VZ is what is called "a full on screen editor". After <LISTing> a programme, READY and flashing cursor appear, you can then <RUN> or EDIT it.

All that is necessary to EDIT it is to move the cursor around anywhere on the screen and type, <INSERT> or <RUBOUT> character/s. Then <RETURN>. With some computers of the very well known variety, you have to call up the line to edit, or go to an EDIT mode.

With a "FRUITY" compatable that I work on, it is quite a pain. The cursor is moved up to the first digit of the line number of the line that is to be edited, then type over the correction, or re-type the whole line. If there are more characters on the line which must remain, then the cursor must be run to the end of the line and only then is the <RETURN> key typed. If not the characters to the end of the line are erased from memory. The cursor is moved around on the screen by pressing other control keys. YUK, what an effort.

So I stress that the VZ is one of the few MOST EASILY EDITED MACHINES. To a beginner it is a charm.

If the programme has a few lines which are similar then rather than type the lines fully, here is a short cut method.

Say the programme has a menu something like this:-

```
100IFX=1THEN5000
110IFX=2THEN6000
   etc.
```

then type line 100 only, <LIST> then move the cursor onto line 100 and change the line number to "110". Change the "1" to "2" and "5000" to "6000", and <RETURN>.

<LIST> again and you will have the two lines, 100 and 110.

In large programmes there could be many lines that are very similar, so much time can be saved with this method.

REMARKS.

Use a good sprinkling of <REMark> statements in your programmes to describe what various parts are for. The VZ will not accept graphic symbols in a <REMark> line unless they are enclosed in quotation marks, thus:-

260 REM"SCORE  "

SPACES.

To indicate a space in a filename or programme when writing it by hand, use a symbol that is not used by the VZ. I use a horizontal squigle "~". So for a filename I write thus:-

CSAVE"WORD GAME~1"

TAPE SAVING.

Another time saver when you have just <CSAVEd> a programme and you wish to <VERIFY> it, IE. CSAVE"CIRCLES 4"

Move the cursor up onto C of CSAVE, do one insert(<CTRL><INSERT>) then VERIFY <CTRL><VERIFY><RETURN>.

The screen should be:-

VERIFY"CIRCLES 4"

. That not only saves time but ensures that you have entered the EXACT filename into the VZ.

Of course a programme can be <VERIFIed> without giving a filename, but the VZ will try to verify the first programme on the tape it receives.

LINE NUMBERS.

A beginner should type in the line numbers as they are in the <LISTing> and not change them. This is because there may be <GOTO> and <GOSUB> statements in the programme, and if you change a line number say from :-

```
5500INPUT"PRESS RETURN TO CONTINUE";Q$
```

to say:-

```
5580INPUT"PRESS RETURN TO CONTINUE";Q$
```

and if there is a line :-

```
7305GOTO5500
```

you will get the error message on the screen :-

```
UNDEF'D STATEMENT IN LINE 7305.
```

As you become more experienced, you can change line numbers. There will be times when you will need to fit more statements in a section of a programme, but there are no more line numbers to use.

IE., you have used all the line numbers from 4560 to 4575, but have to put a statement in line 4570. You then have to make line 4570 -> 4571, 4571 -> 4572 etc. You then have to change any <GOTO> and <GOSUB> statements to suit.

This is easy with small programmes, but it's a different situation with large ones. The statement/command called "RENUMBER" in the EXTENDED BASIC unit will do this for you, by changing line numbers and <GOTO>/<GOSUB> numbers automatically.

If you are writing your own programme, I suggest that the first line number be 1000. The various "blocks" of the programme should be in multiples of 1000. So The MENU could commence on 1000 and other "blocks" at 2000, 4000, 5000, 6000, 10000 etc.

By not doing this and starting at line 10, you will soon find that there are not enough line numbers at the start to add other sections to it.

You can use the AUTO line number option in the EXTENDED BASIC or this simple method to automatically set the starting line number and increment value.

On line 0 (zero) type,
REM1000,20 :-
0REM1000,20
Now without typing a line number, type in immediate mode:- POKE 31469,183<RETURN>

This sets the VZ in AUTO LINE NUMBER mode.
Now <RUN><RETURN>
and the screen will show 1000 with cursor ready for you to type the statement. After <RETURN> the next line number will be 2020.
The increments will be by 20. To start at 4500 in increments of 10, then line :- 0REM4500,10

To RUN your programme, <RUN>1000 or whatever the commencing line is. To continue in AUTO mode just <RUN><RETURN>. The first line with statement will show and can be edited if required or left as is :- <RETURN>. The next line will show and so on. When you are finished with AUTO just erase line 0:- 0<RETURN>

If you want AUTO back again, type 0 and the POKE as before.

DELETE.

To delete a line just type the line number and <RETURN>. If there are lots of consecetive lines to erase this is a quick method. DELETE is another EXTENDED BASIC command, but it can be implimented just as easily as the AUTO command.

Type OD2300-3000<RETURN>

POKE31469,182<RETURN>
<RUN><RETURN>

Lines 2300 to 3000 will be deleted. So set the two numbers on line O to suit. When finished, erase line O.

HINTS.

A comma "," can be typed instead of "THEN" in an "IF THEN" statement.

A question mark "?" can be typed instead of "PRINT" in a PRINT statement.

An apostrophe "'" can be typed instead of a "REM" in a REM statement.

In a SOUND statement, it is not necessary to type thus:-
SOUND15,5:SOUND18,3:SOUND20,1

as the short method is thus:-

SOUND15,5;18,3;20,1

note the semicolon ";".

COMMUNICATIONS ADDRESSES.

78FDH & 78FEH	the starting address of free space in RAM.
78F6H & 78F7H	last line number excecuted.
78E2H & 78E3H	starting line number.
7899H	single byte, last key pressed.
789EH	single byte, high or low res.
789AH	single byte, error code storage.
78A2H & 78A3H	current line number.
78A7H & 78A8H	address of the start of the keyboard buffer.
78D6H & 78D7H	address of the next available location in the string area.
78DAH & 78DbH	line number of the last DATA statement read.
7921H & 7922H	USR argument address.
7815H 0	disable keyboard.
7818H 1	inverse VDU.

0741d

NOW SOME PROGRAMMING HINTS.

This short routine is similar to the AUTO and DELETE one discussed elsewhere. line 500 must be the first line of your programme. 218 is the TOKEN POKEd to give free memory in number of bytes EI. FRE(O)

```
500 PRINTPRINT(O)
510 REM LINE 500 "FRE(O)" IS POK
ED BY 31470,218
POKE31470,218
```

Use it to give some indication of free available memory while you are writing a large programme.

TRON AND TROFF.

This is used to "trace" a programme from line number to line number. It prints on the VDU. the line numbers in horizontal vees IE. <3005>. If there is text or graphics on the VDU., the line numbers will of course print over the top of those.

POKE31003,175 enables TRON.

POKE31003,0 disables (switches off) TROFF.

This will print on the VDU. or printer the characters after the CHR$(13) part of the statement, on the next line. The same as a Carriage Return.

```
50 REM PRINTS CHARACTERS ON NEXT LINE
100 PRINT"ABC";CHR$(13);"123"
```

```
ABC
123
```

This routine inverses the INPUT statement on the VDU. and also PRINTs in inverse. This is acheived by line 70, then dis-enabled by line 100.

```
3 REM INVERSE INPUT AND PRINT
5 CLS
10 PRINT"START"
50 INPUT"ENTER NAME ";Q$
70 POKE30776,10:INPUT"AGE ";A$
80 PRINT"NAME ";Q$
90 PRINT"AGE ";A$
100 POKE30776,1
200 INPUT"TIME ";T$
220 PRINT"TIME ";T$
```

This routine inverses the PRINT of a $string on the VDU. and a printer, if it is programmed to do so. Line 180 with OR statement enables it, and line 220 with the AND statement dis-enables it.

```
50 CLS
100 REM TO INVERSE A STRING WITHIN A PROGRAMME.
120 A$="TEST PROGRAMME"
130 B=15432
150 PRINTA$:PRINTB
160 PRINT"----------------------"
180 POKE30776,PEEK(30776)OR2
200 PRINTA$:PRINTB
220 POKE30776,PEEK(30776)AND253
260 PRINT"----------------------"
280 PRINTA$:PRINTB
```

```
TEST PROGRAMME
 15432
----------------------
TEST PROGRAMME
 15432
----------------------
TEST PROGRAMME
 15432
```

Variation to INKEYS.

INKEYS is used to allow entry of a key without having to press the <RET key. In a menu if a letter is asked for the instructions are thus....

```
5 CLS
10 REM  VARIATION TO "INKEY$" CONVERT TO ASCII FOR MENU SELECT
50 PRINT"A = AAA"
55 PRINT"B = BBB"
60 PRINT"C = CCC"
90 PRINT"TYPE IN A - C FOR SELECTION"
100 A$=INKEY$
110 A$=INKEY$:IFA$=""THEN100
120 AS=ASC(A$)
130 IFAS=65THENPRINT"YOU SELECTED AAA":END
135 IFAS=66THENPRINT"YOU SELECTED BBB":END
140 IFAS=67THENPRINT"YOU SELECTED CCC":END
145 IFAS>67ORAS<65THENPRINT"SELECT AGAIN":GOTO100
```

This short routine flashes "C" on the VDU. waiting for the "C" key to be pressed so that the programme can continue.

```
10000 REM FLASHING " C "
10005 PRINT@485,"PRESS <C> TO CONTINUE";
10010 PRINT@492,"C";
10015 FORT=1TO500:NEXT
10040 PRINT@492," ";
10045 FORT=1TO500:NEXT
10050 GOTO10000
10060 END
10070 GOTO10010
```

This one will allow a BMC BX-80 printer to work from the COPY command, for HI-RES or LO-RES.

```
100 REM OPERATE BMX BC-80 PRINTER IN COPY MODE
1000 LPRINTCHR$(15);
1010 LPRINTCHR$(27);"A";CHR$(6);
1020 FORY%=0TO63
1030 FORX%=0TO127
1040 P=POINT(X%,Y%)
1050 IFP=1THENLPRINT" ";:NEXT:GOTO1070
1060 LPRINT"*";:NEXT
1070 LPRINT:NEXT
```

This one flashes the message "**** STOP TAPE ****" on the VDU.

```
10 CLS
20 FORL=1TO6
30 PRINT@230,"**** STOP TAPE ****"
50 SOUND8,4
60 PRINT@230,"                    "
```

JOYSTICK DRAWER

```
10 MODE(1)
20 X=0
30 Y=0
40 A=(INP(43)AND31)
50 IFA=23ANDX<127THENX=X+1
60 IFA=27ANDX>0THENX=X-1
70 IFA=30ANDY>0THENY=Y-1
80 IFA=29ANDY<63THENY=Y+1
90 SET(X,Y)
100 GOTO40
```

BASIC DODGE

```
5 POKE30744,1:' IF YOU HAVE A EARLIER VZ
 YOU DO NOT NEED THE POKE
6 CLS
10 A=28672:X=16
20 I$=INKEY$:IF I$="K"THENX=X-1
30 IFI$="L"THENX=X+1
40 IFPEEK(A+X)<>32THEN200
50 PRINT@X,"V";:S=S+1
60 PRINT@480+RND(31),"*"
70 GOTO 20
200 CLS
210 SOUND1,1:PRINT"GAME OVER ♀ ♀ ♀"
220 PRINT"SCORE=";S
230 IF INKEY$="S"THEN RUN ELSE 230
```

To give you a gentle start, here are four very short programmes contributed by Larry Taylor.

The first draws a circle, the second a triangle, the third a spiral and the fourth a star.

You can experiment with these to give different results.

```
15 MODE(1):COLOR3
20 R=6.3
30 FORA=0TO30STEP.02
40 X=64+7*R*COS(A)
50 Y=32+5*R*SIN(A)
60 SET(X,Y)
70 NEXTA
80 GOTO80
```

```
10 MODE(1):COLOR2
20 FORI=99TO0STEP-1
30 SET(I,I/2)
31 NEXTI
34 FORK=1TO50
35 SET(K/2,K)
36 NEXTK
40 FORT=25TO100
45 SET(T,50)
50 NEXTT
80 GOTO80
```

```
10 CLS
15 MODE(1)
20 FORA=0TO30STEP.02
30 R=A*.3:IFR>6.8THENGOTO60
40 SET(64+7*R*COS(A),33+5*R*SIN(A))
50 NEXTA
60 GOTO60
```

```
10 CLS
15 MODE(1)
20 FORA=0TO30STEP.02
30 R=6*COS(2*A/3)
40 SET(64+7*R*COS(A),33+5*R*SIN(A))
50 NEXTA
60 GOTO60
```

Two more from Larry Taylor.

The first draws knots and the second a flower.

```
10 CLS
15 MODE(1):COLOR8,1
20 FORA=0TO30STEP.02
30 R=A*COS(A)*SIN(A):IFR>11THENGOTO60
40 SET(64+7*R*COS(A),32+3*R*SIN(A))
50 NEXTA
60 GOTO60
```

```
10 CLS
15 MODE(1)
20 FORA=0TO30STEP.02
30 R=6*COS(3*A/2)
40 SET(64+7*R*COS(A),33+5*R*SIN(A))
50 NEXTA
60 GOTO60
```

This one called NAME is from Jamie Perry of Dick Smith Electronics in Sydney.

```
1 CLS
5 DIMB$(40)
10 PRINT"HELLO MY NAME IS VZ-300"
20 INPUT"WHAT IS YOUR NAME (FIRST&LAST)";A$:IFA$=""THEN20
22 L=LEN(A$)
30 PRINT:PRINT:PRINT"THANKYOU ";
40 FORI=1TOL:B$(I)=MID$(A$,I,1):NEXTI
50 FORI=LTO1STEP-1:PRINTB$(I);:NEXTI
60 PRINT".":PRINT"OOPS I GUESS I GOT IT BACKWARDS"
70 PRINT"A SMART COMPUTER LIKE ME SHOULD"
72 PRINT"NOT MAKE A MISTAKE LIKE THAT!"
80 PRINT"BUT I JUST NOTICED YOUR LETTERS"
82 PRINT"ARE OUT OF ORDER."
90 PRINT"LETS PUT THEM LIKE THIS: "
100 FOR J=2 TO L:I=J-1:T$=B$(J)
110 IF T$>B$(I)THEN 130
120 B$(I+1)=B$(I):I=I-1:IFI>0THEN110
130 B$(I+1)=T$:NEXTJ
140 FORI=1TOL:PRINTB$(I);:NEXT:PRINT:PRINT
150 INPUT"DON'T YOU LIKE THAT BETTER";D$
160 IFD$="YES"THEN180
170 PRINT:PRINT"I'M SORRY YOU DON'T LIKE IT":GOTO200
180 PRINT:PRINT"I KNEW YOU'D AGREE!!"
200 PRINT:PRINT"I REALLY ENJOYED MEETING YOU"
210 PRINTA$;"  HAVE A NICE DAY"
```

The following programmes are all interesting so type them in. The REM statement lines should give some indication of what the programmes are about.

```
10 REM ++SONG++
20 CLS
30 INPUT"ENTER NO.OF NOTES";N
40 PRINT"ENTER YOUR NOTES"
50 DIM A%(2*N-1)
60 FOR I=0TO N-1
70 INPUT"FREQ CODE 15 TO 31";A%(I*2)
80 INPUT"DURATION CODE 1 TO 7";A%(I*2+1)
90 NEXT
100 FORI=0TON-1
110 SOUND A%(I*2),A%(I*2+1)
120 NEXT
```

```
10 REM BOUNCING NAME
15 CLS
20 A=6:B=11
30 Y=1:X=1
40 EA=A:EB=B
45 FORG=1TO 80
60 PRINT@(32*B+A),"LEA MATHEWS"
70 B=B+X
80 A=A+Y
90 IFA<2THENY=-Y
100 IFA>30THENY=-Y
110 IFB<2THENX=-X
120 IFB>14THENX=-X
125 FOR T=1TO50
130 IF G= 80 THEN 10
140 NEXT G
145 GOTO 40
```

```
4 COLOR,0
5 SOUND25,6:SOUND10,6
10 REM HEX TO DECIMAL
15 CLS
20 INPUT"ENTER FOUR DIGIT HEX NO.";N$
25 IFN$="S"THEN END
27 IFLEN(N$)<>4THEN20
30 A$=MID$(N$,1,1)
40 B$=MID$(N$,2,1)
50 C$=MID$(N$,3,1)
55 D$=MID$(N$,4,1)
60 E$=A$:GOSUB200:A=E*16^3
70 E$=B$:GOSUB200:B=E*16^2
80 E$=C$:GOSUB200:C=E*16
90 E$=D$:GOSUB200:D=E
100 PRINT
110 PRINTN$;"HEX =";A+B+C+D;"DECIMAL"
120 PRINT"________________________________"
130 GOTO20
200 IFVAL(E$)<10THENE=VAL(E$)
205 IFE$="A"THENE=10
210 IFE$="B"THENE=11
215 IFE$="C"THENE=12
220 IFE$="D"THENE=13
225 IFE$="E"THENE=14
230 IFE$="F"THENE=15
235 RETURN
```

```
2 REM RANDOM SOUND AND COLOUR
5 CLS
10 SOUNDRND(31),RND(9)
30 COLOR,0
35 SOUNDRND(31),RND(9)
36 COLOR,1
40 GOTO10
```

```
60000 REM DEC TO HEX
60005 CLS
60010 INPUT"DEC VALUE";BY
60020 IFBY>255THENPRINT"TOO BIG":GOTO60010
60025 GOSUB60100
60030 PRINT"DEC";BY" IS HEX ";A$
60031 PRINT"-----------"
60035 GOTO60010
60100 REM HEX TO DEC
60110 TA$="0123456789ABCDEF":A$=""
60120 H1=INT(BY/16)+1
60125 H2=BY-16*(H1-1)+1
60130 A$=MID$(TA$,H1,1)+MID$(TA$,H2,1)
60150 RETURN
```

```
10 REM V-WING SPACE BATTLE
20 CLS
30 SC=0
100 FORZ=1TO 20
110 SL=28736:M=22:D=-32
120 POKE28715+INT(RND(0)*468),INT(RND(0)*9)+49
130 W=SL
140 A$=INKEY$
150 IFA$=","THENSL=SL+1:M=62
160 IFA$="M"THENSL=SL-1:M=60
170 IFA$="."ANDSL>28736THENSL=SL-32:M=1
180 IFA$=" "ANDSL<29151THENSL=SL+32:M=22
190 Q=PEEK(SL)
200 IFQ>48ANDQ<58THENSC=SC+Q:GOTO980
205 POKEW,32
210 POKESL,M
220 IFRND(0)<.99THEN130
230 COLOR,1:FORT=1TO 20:NEXTT:COLOR,0
980 CLS:PRINT@0,"SCORE    ";SC;"    ";20-Z;"SHIPS LEFT"
982 COLOR,INT(RND(0)*2)
985 SOUNDRND(0)*25+1,1:IFRND(0)>.6THEN985
990 NEXTZ
1000 PRINT@0,"THE BATTLE IS OVER","YOU SCORED   ";SC
1100 COLOR,INT(RND(0)*2)
1120 GOTO1100
1200 END
```

```
5 REM TEST ONE JOYSTICK
10 CLS
20 A=(INP(43)AND31)
30 IFA=30THENPRINT"UP":GOTO20
40 IFA=29THENPRINT"DOWN":GOTO20
50 IFA=27THENPRINT"LEFT":GOTO20
60 IFA=23THENPRINT"RIGHT"
70 GOTO20
```

TEST JOYSTICKS.

The first is to test one only Joystick. The second one is to test two Joysticks.
These can be the basis of games or drawing Programmes. Elsewhere in the book is an ASSEMBLY listing routine that will of course run faster.

```
1 REM TEST TWO JOYSTICKS
5 R$="RIGHT JOYSTICK  ":L$="LEFT JOYSTICK "
7 CLS
10 A=INP(32)AND31:IFA=31THEN10:REM WAIT FOR SOME ACTION
20 A=INP(46)AND31:IFA=31THEN100:REM CHECK FIRST ROW
30 IFA=26THENPRINTR$+"LEFT+UP":GOTO200
32 IFA=25THENPRINTR$+"LEFT+DOWN":GOTO200
34 IFA=22THENPRINTR$+"RIGHT+UP":GOTO200
36 IFA=21THENPRINTR$+"RIGHT+DOWN":GOTO200
40 IFA=30THENPRINTR$+"UP":GOTO200
50 IFA=29THENPRINTR$+"DOWN":GOTO200
60 IFA=27THENPRINTR$+"LEFT":GOTO200
70 IFA=23THENPRINTR$+"RIGHT":GOTO200
80 IFA=15THENPRINTR$+"ARM":GOTO200
100 A=INP(45)AND16:REM NOW CHECK SECOND ROW
110 IFA=0THENPRINTR$+"FIRE":GOTO200
120 A=INP(43)AND31:IFA=31THEN190:REM CHECK 3RD ROW
130 IFA=26THENPRINTL$+"LEFT+UP":GOTO200
132 IFA=25THENPRINTL$+"LEFT+DOWN":GOTO200
134 IFA=22THENPRINTL$+"RIGHT+UP":GOTO200
136 IFA=21THENPRINTL$+"RIGHT+DOWN":GOTO200
140 IFA=30THENPRINTL$+"UP":GOTO200
150 IFA=29THENPRINTL$+"DOWN":GOTO200
160 IFA=27THENPRINTL$+"LEFT":GOTO200
170 IFA=23THENPRINTL$+"RIGHT":GOTO200
180 IFA=15THENPRINTL$+"ARM":GOTO200
190 A=INP(39)AND16:REM CHECK 4TH ROW
195 IFA=0THENPRINTL$+"FIRE"
200 FORI=1TO300:NEXTI:GOTO10
```

```
1 GOTO5
2 BSAVE"12345678",7000,7800
3 END
4 BLOAD"12345678":GOTO50
5 FORU=-28707TO-28674
6 READ W:POKEU,W:NEXT
7 CLS:INPUT"[illegible]RAW OR [illegible]OAD PICTURE";C$
10 CLS:PRINT"[illegible]";
11 PRINT"[illegible]
12 PRINT"[illegible] SELECTS COLOUR
13 PRINT"[illegible] TRANSPERANT PEN
14 PRINT"[illegible] RANDOM COLOURS"
15 PRINT"[illegible] SELECTS PEN WIDTH"
16 PRINT"[illegible] SAVE PICTURE TO DISK
17 PRINT"[illegible] INVERSE SCREEN"
18 PRINT"[illegible] CLEAR SCREEN":PRINT"[illegible] STORE BLOCK"
19 PRINT"[illegible] DRAW CIRCLE(SOLID)"
20 PRINT"[illegible]";
23 PRINT@386,"SAVING NAME(8)";:INPUTV$
24 IFLEN(V$)<8ORLEN(V$)>8THENSOUND2,1:GOTO23
25 IFC$="L"THEN550
26 PRINT@386,:INPUT"X COORDINATE(1-127)";X
27 IFX<1ORX>127THENSOUND1,8:GOTO26
28 PRINT@386,:INPUT" Y COORDINATE (1-63)";Y
29 IFY<1ORY>63THENSOUND1,8:GOTO28
30 PRINT@386,:INPUT"BACKGROUND COLOUR(1TO4) ";D
40 IFD<1ORD>4THEN30
43 MODE(1):IFD=1THEN50ELSE300
50 DIMP%(10,8):W=1:B=1:R=1:T=1:C=1:S=1
55 IFX=0THEND=1:X=64:Y=32
100 C$=INKEY$:C$=INKEY$
110 IFC$="Z"THENSOUND25,1:GOTO450
120 IFC$="M"THENX=X-1
125 IFC$=","THENX=X+1
130 IFC$="."THENY=Y-1
140 IFC$=" "THENY=Y+1
145 IFC$="I"THENY=Y-1:X=X-1
150 IFC$="O"THENX=X+1:Y=Y-1
155 IFC$="L"THENX=X+1:Y=Y+1
160 IFC$="K"THENX=X-1:Y=Y+1
161 IFX>126THENX=X-1:SOUND1,2
162 IFX<1THENX=X+1:SOUND1,2
163 IFY>62THENY=Y-1:SOUND1,2
164 IFY<1THENY=Y+1:SOUND1,2
172 IFC$="C"ORC$="S"THENGOTO350
175 IFC$="6"THENR=R*-1:SOUND23,1
180 IFVAL(C$)>0ANDVAL(C$)<5THENC=VAL(C$):SOUND29,1
182 IFR=-1THENC=RND(4)
183 IFC$="8"THENSOUND18,1:GOTO500
184 IFC$="9"THENB=B*-1:SOUND31,1
185 IFB=-1THENCOLOR,1ELSECOLOR,0
187 IFW=-1ANDT=1THENSOUND2,4
190 IFC$="5"THENT=T*-1:SOUND28,1
193 IFC$="7"THENW=W*-1:SOUND20,1
194 IFC$="-"THENSOUND24,1:GOTO400
195 E=POINT(X,Y):COLORE+1:SET(X,Y)
196 IFC$="0"THENSOUND10,2:RUN30
199 FORA=1TO100:IFJ=31THENNEXT
200 IFT=1THENSET(X,Y):COLORE:SET(X,Y)ELSECOLORC:SET(X,Y)
205 IFW=-1THEN250ELSE100
250 FORA=-1TO1
```

```
253 FORG=-1TO1
255 SET(X+A,Y+G)
270 NEXT:NEXT:GOTO100
280 GOTO100
300 POKE30862,241:POKE30863,143
305 DATA 33,0,112,17,179,132,1,0,8,26,119,35,19,11,120,177,194
307 DATA230,143,201,33,0,112,17,1,112,1,255,7,54,85,237,176,201
315 IFD=3THENPOKE(-28677),170ELSEPOKE(-28677),85
316 IFD=4THENPOKE(-28677),255
320 X=USR(X):GOTO50
350 SOUND22,1:G=Y
351 K$=INKEY$:K$=INKEY$:IFK$="X"ANDG>YTHENG=G-Y:COLORC:GOTO360
352 IFK$="V"THENG=G+1
353 E=POINT(X,G):COLORE+1:SET(X,G)
354 COLORE:SET(X,G)
355 IFG=63THEN350ELSE:GOTO351
360 FORA=0TO6.3STEP(.7/G):H=(SIN(A)*(1.5*G)+X):I=(COS(A)*G+Y)
365 IFH>126ORI>62THENSOUND2,3:GOTO100
370 SET(H,I):NEXT:IFG=1ORC$="C"THEN100ELSEG=G-.5:GOTO360
400 COLORD:IFX<4ORX>122ORY<3ORY>59THENSOUND3,4:GOTO100
402 FORA=1TO10:FORG=1TO8
410 P%(A,G)=POINT(X+A-5,Y+G-4):SET(X+A-5,Y+G-4):NEXT:NEXT
420 SOUND24,1:GOTO100
450 IFX<5ORX>122ORY<3ORY>59ORP%(1,1)=0THENSOUND3,4:GOTO100
453 FORA=1TO10:FORG=1TO8
460 COLORP%(A,G):SET(X+A-5,Y+G-4):NEXT:NEXT:GOTO100
500 FORA=1TO8
510 POKE31481+A,ASC(MID$(V$,A,1))
520 NEXTA
530 GOTO2
550 CLS:INPUT"NAME OF PICTURE";C$
555 IFLEN(C$)<8ORLEN(C$)>8THENSOUND2,1:GOTO550
560 FORA=1TO8
570 POKE31517+A,ASC(MID$(C$,A,1))
580 NEXTA
590 MODE(1):GOTO4
```

This programme requires a Disc System. Note the DATA statement lines 305 and 307. The DATA is of course in decimal, which represents HEX values of a Machine Language routine.

```
5 D=1:REM*SORT VIA KB*
7 CLEAR500:CLS
8 PRINT"TO FINISH ENTRY, TYPE (END)":PRINT
10 INPUT"NEXT NAME";A$(D)
12 IF A$(D)="END" GOTO 30
20 D=D+1
21 N=N+1
22 GOTO 10
30 FOR F=1TO N-1
40 FOR S=F+1 TO N
50 IF A$(F)<=A$(S) THEN90
60 T$=A$(F)
70 A$(F)=A$(S)
80 A$(S)=T$
90 NEXT S
100 NEXT F
110 FOR D=1 TO N
111 PRINT A$(D)
112 NEXT D
```

This SORT VIA KEYBOARD Programme introduces a sort function. It sorts alphabeticly A to Z. Type "END" when you have finished typing in the names.

```
10 REM PYRAMIDS
20 CLS:INPUT"PYRAMID HEIGHT NO HIGHER THAN 60";H
22 INPUT"LENGTH OF BASE NO HIGHER THAN 63";B
25 D=B/2
30 IFB<1ORB>63ORH<0ORH>60THEN20
40 CLS:MODE(1):COLOR6,1:REM CYAN
50 DL=(63-B)+(B/2.5)
55 DU=60-H:DM=63-B
57 DX=60-INT(H/2.5)
60 Y1=DU:X1=DL:Y2=60:X2=63+D:GOSUB1000
65 DX=60-INT(H-2.5)
70 Y1=60:X1=DM:GOSUB1000
80 Y1=DX:Y2=DX:GOSUB1000
90 FORZ=Y1TO60:SET(X1,Z)
95 SET(X2,Z):NEXTZ
100 X2=DL:Y1=60:Y2=DU:GOSUB1000
110 Y1=DX:GOSUB1000
120 X1=63+D:GOSUB1000
130 COLOR7,1
140 DN=63+B/2:DK=(63+B/2)-(B/2.5)
150 X2=DK:X1=DN:GOSUB1000
160 X1=63-B:GOSUB1000
170 Y1=60:GOSUB1000
180 X1=DN:GOSUB1000
190 FORZ=1TO5000:NEXTZ
200 INPUT"AGAIN";A$
210 IFLEFT$(A$,1)="Y"THEN20
220 END
1000 S=1:IFX1>X2ANDY1>Y2THENS=-1
1010 SET(X1,Y1):SET(X2,Y2)
1015 Y=Y1:N=1:IFY1=Y2THENA1=0:GOTO1030
1020 A1=(X2-X1)/(Y2-Y1):IFS=-1THENA1=-A1
1030 FORX=X1TOX2STEPS
1035 IFX<0THENX=0
1040 IFY<0THENY=0
1050 SET(X,Y):N=N+1
1060 IFA1<>0THENY=Y1+N/A1
1070 NEXTX:RETURN
```

```
1 CLS:POKE30744,1
2 CLEAR100
3 PRINT@200,"[graphics]"
4 PRINT@232,"[graphics]"
5 PRINT@264,"[graphics]"
6 PRINT@296,"SELECT YOUR HORSE.":PRINT@328,"CHOOSE [graphics],[graphics],[graphics],[graphics]OR [graphics]."
7 FORT=1TO4000:NEXT:CLS
8 P=28672
10 PRINT@0,"[graphics]";:PRINT@32,"[graphics]";:PRINT@64,"[graphics]";:PRINT@96,"[graphics]"
15 PRINT@128,"[graphics]";:PRINT@160,"[graphics]";:PRINT@192,"[graphics]";
16 PRINT@224,"[graphics]";:PRINT@256,"[graphics]"
20 PRINT@30,"[graphics]";:PRINT@62,"[graphics]";:PRINT@94,"[graphics]";:PRINT@126,"[graphics]"
22 PRINT@158,"[graphics]";:PRINT@190,"[graphics]";:PRINT@222,"[graphics]";:PRINT@254,"[graphics]";
23 PRINT@286,"[graphics]";
25 FORM=298TO318:POKEP+M,220:NEXT
40 FORV=33TO61:POKEP+V,45:NEXT:FORW=97TO125:POKEP+W,45:NEXT
42 FORQ=161TO189:POKEP+Q,45:NEXT:FORK=225TO253:POKEP+K,45:NEXT
43 PRINT@320,"[graphics]":FORT=1TO1500:NEXT
44 PRINT@320,"                    "
45 A=1:C=65:E=129:G=193:H=257
50 Z=29
55 POKEP+A,32:POKEP+C,32:POKEP+E,32:POKEP+G,32:POKEP+H,32
60 X=INT(RND(5))
65 IFX=1THENA=A+1
70 IFX=2THENC=C+1
75 IFX=3THENE=E+1
80 IFX=4THENG=G+1
81 IFX=5THENH=H+1
82 FORN=1TO10
83 NEXTN
85 PRINT@A,"[graphics]":PRINT@C,"[graphics]"
90 PRINT@E,"[graphics]":PRINT@G,"[graphics]"
91 PRINT@H,"[graphics]"
95 IFA=ZORC=Z+64ORE=Z+128ORG=Z+192ORH=Z+256THEN100ELSE150
100 IFA=ZTHENGOSUB200ELSE105
102 PRINT1:GOSUB260
104 GOTO43
105 IFC=Z+64THENGOSUB200ELSE110
107 PRINT2:GOSUB260
109 GOTO43
110 IFE=Z+128THENGOSUB200ELSE115
112 PRINT3:GOSUB260
114 GOTO43
115 IFG=Z+192THENGOSUB200ELSE120
117 PRINT4:GOSUB260
119 GOTO43
120 IFH=Z+256THENGOSUB200
122 PRINT5:GOSUB260
124 GOTO43
150 GOTO55
200 PRINT@320,"[graphics] ";
205 RETURN
260 PRINT@384,"IF YOU WISH TO SEE ANOTHER"
265 PRINT@416,"RACE, PRESS [graphics]. IF YOU DON'T"
270 PRINT@448,"THEN PRESS ANY KEY"
275 AN$=INKEY$
280 AN$=INKEY$:IFAN$=""THEN280
285 IFAN$="R"THEN290ELSECLS:END
290 FORL=1TO29:POKEP+L,32:NEXT
295 FORL=65TO93:POKEP+L,32:NEXT
300 FORL=129TO157:POKEP+L,32:NEXT
```

```
310 FORL=193TO221:POKEP+L,32:NEXT
320 FORL=257TO285:POKEP+L,32:NEXT
325 FORL=320TO334:POKEP+L,32:NEXT
330 FORL=352TO384:POKEP+L,32:NEXT
340 FORL=384TO416:POKEP+L,32:NEXT
350 FORL=416TO448:POKEP+L,32:NEXT
360 FORL=448TO480:POKEP+L,32:NEXT
370 RETURN
```

```
10 CLS
20 PRINT "DAY OF THE WEEK"
30 PRINT
40 PRINT "(ENTER 0,0,0 TO END PROGRAM)"
50 PRINT "MONTH, DAY, YEAR";
60 INPUT M,D,Y
70 IF M<>0 THEN 110
80 IF K<>0 THEN 110
90 IF Y<>0 THEN 110
100 GOTO 370
110 IF M>2 THEN 140
120 M=M+12
130 Y=Y-1
140 N=D+2*M+INT(.6*(M+1))+Y+INT(Y/4)-INT(Y/100)+INT(Y/400)
150 N=INT((N/7-INT(N/7))*7+.5)
160 IF N>0 THEN 190
170 PRINT "SATURDAY"
180 GOTO 350
190 IF N>1 THEN 220
200 PRINT "SUNDAY"
210 GOTO 350
220 IF N>2 THEN 250
230 PRINT "MONDAY"
240 GOTO 350
250 IF N>3 THEN 280
260 PRINT "TUESDAY"
270 GOTO 350
280 IF N>4 THEN 310
290 PRINT "WEDNESDAY"
300 GOTO 350
310 IF N>5 THEN 340
320 PRINT "THURSDAY"
330 GOTO 350
340 PRINT "FRIDAY"
350 PRINT
360 GOTO 50
370 END
```

```
1 POKE30744,1:CLS:PRINT"  [illegible] BY JAMIE PERRY 1984":PRINT
2 PRINT"  .   =  20 FUEL CELLS"
3 PRINT"  +   =  50 FUEL CELLS"
4 PRINT"  *   =  INSTANT DEATH"
5 PRINT"  V   =  YOU":PRINT
6 PRINT"  M   =  MOVE LEFT"
7 PRINT"  ,   =  MOVE RIGHT"
8 PRINT"  S   =  START":PRINT:PRINT"  HINT\ WATCH YOUR FUEL"
9 FORC=1TO5000:IFINKEY$="S"THEN10ELSENEXT
10 CLS
50 A=28850:S=100:T=1:A$=""
100 PRINT@480+RND(26),"* .";A$
101 IFT/100=INT(T/100)THENA$=A$+" *":PRINT@99,"[illegible]":SOUND1,2
102 J=PEEK(A):IFJ=42THEN200
103 IFJ=46THENSOUND30,1:S=S+20:POKEA+1,41:POKEA-1,40
104 IFJ=43THENCOLOR,1:SOUND29,1;25,1:S=S+50:COLOR,0
105 POKEA,22
106 IFRND(99)>90THENPRINTTAB(RND(29));"+";
107 S=S-2:PRINT  @0,"[illegible]";S:T=T+1
108 IFS=0THENPRINT@200,"[illegible]":GOTO200
125 POKEA,32
130 IFC<5001THEN140ELSE152
140 IFINKEY$="M"THENA=A-1:POKE26666,1:POKE26666,0
150 IFINKEY$=","THENA=A+1:POKE26666,1:POKE26666,0
151 GOTO100
152 IFPEEK(A+63)=46ORPEEK(A+63)=43ORPEEK(A+94)=46THENA=A-1
153 IFPEEK(A+65)=46ORPEEK(A+65)=43ORPEEK(A+98)=46THENA=A+1
154 IFT<HANDPEEK(A+32)=42THENA=A+1
155 IFINKEY$="S"THENC=0:GOTO10
160 GOTO100
200 POKEA,24
205 POKE30744,0
210 PRINT@300,"[illegible]";T
211 IFT>HTHENH=T
212 PRINT@364,"[illegible]";H:IFH=TTHENPRINT@352,"[illegible]"
213 IFH=TTHENSOUND25,4;22,3;29,2;31,1;29,2;27,3;24,2;29,3
214 IFH=TTHENSOUND0,9;0,9:GOTO218
215 PRINT@396,"[illegible]";N$;"[illegible]"
216 SOUND16,5;0,1;16,5;0,1;16,2;16,1;19,5
217 SOUND18,4;18,3;16,4;16,3;15,4;16,4
218 POKE30744,1:IFC=5001THENN$="V-ZED":GOTO220
219 IFH=TTHENCLS:INPUT"NAME PLEASE";N$:GOTO1
220 FORA=1TO1000
221 IFINKEY$="S"THEN10
222 NEXT:GOTO1
```

```
5 REM **********************
6 REM **   SOUND EFFECTS    **
7 REM ** BY ANDREW WILLOWS **
8 REM **********************
9 CLS
10 FORT=-28687TO-28676
20 READD:POKET,D:NEXT
30 DATA 229,033,160,000,001,003,000,205,092,052,225,201
40 POKE30862,241:POKE30863,143
45 REM**"[illegible]"**
46 PRINT"   DECAYING ZOOP"
50 FORT=1TO255STEP4:POKE-28685,T:X=USR(0):NEXT
55 SOUND0,4
56 REM**"[illegible]"**
57 PRINT"   INCREACING ZOOP"
60 FORT=255TO1STEP-4:POKE-28685,T:X=USR(0):NEXT
65 SOUND0,4
66 REM**"[illegible]"**
67 PRINT"   RANDOM BEEPS"
70 POKE-28682,10
74 FORT=1TO50
75 R=RND(254)+1:POKE-28685,R:X=USR(0)
76 NEXT
77 POKE-28682,70
78 SOUND0,4
79 REM**"[illegible]"**
80 PRINT"   WAVES":POKE-28682,1
85 FORY=1TO10
86 FORT=1TO10:POKE-28685,T:X=USR(0):NEXTT
87 FORT=30TO1STEP-1:POKE-28685,T:X=USR(0):NEXTT
88 NEXTY
89 POKE-28682,4:SOUND0,4
90 REM**"[illegible]"**
91 PRINT"   INCREASING PHASOR":FORY=1TO20
95 FORT=10TO1STEP-1:POKE-28685,T:X=USR(0):NEXTT
96 NEXTY
97 SOUND0,4
98 REM**"[illegible]"**
99 PRINT"   DECREACING PHASOR"
100 FORY=1TO20
105 FORT=1TO10:POKE-28685,T:X=USR(0):NEXTT
106 NEXTY
107 SOUND0,4
108 REM**"[illegible]"**
109 PRINT"   UFO LEAVING"
110 C=61:FORT=60TO1STEP-1
115 POKE-28682,T:POKE-28685,C
120 C=C-1:X=USR(0):NEXT
125 SOUND0,4
126 REM**"[illegible]"**
127 PRINT"   UFO LANDING"
130 C=1:FORT=1TO60
135 POKE-28682,T:POKE-28685,C
140 C=C+1:X=USR(0):NEXT
145 SOUND0,4
146 REM**"[illegible]"**
147 PRINT"   BUZZER"
150 POKE-28682,3:POKE-28685,60
155 FORT=1TO100:X=USR(0):FORY=1TO5:NEXTY:NEXT
160 SOUND0,4
199 POKE-28682,3
200 GOTO50
```

```
2 CLS
3 COLOR7
5 A=INT(RND(19))+1
7 D$="THE ANSWER IS "
8 E$="NO. OF GOES LEFT"
10 FORF=0TO15
15 PRINT"█    █"
20 NEXT
25 PRINT"██████"
27  PRINT@6,"▀ █ █ █ █ ▀▀ ▀ █ ▀▄ █ ▀"
29 PRINT@38,"█ █ █ █ █ █ █ █ █ █ ▀█ ▀"
31 PRINT@70,"▀▀ ▀▀ ▀ ▀▀ ▀▀ ▀▀ ▀ ▀ ▀ ▀ ▀▀"
45 PRINT@257,"-0-"
48 PRINT@37,"-"
50 PRINT@1,"---"
55 PRINT@33,"  /"
60 PRINT@65," /"
65 PRINT@97,"/"
70 N=4
95 FORF=1TO4
100 PRINT@264,"PICK A NUMBER";:INPUTG
105 IFA=GTHEN350
108 N=N-1
110 A$="HIGHER "
115 IFG>ATHENA$="SMALLER"
120 PRINT@296,A$
155 PRINT@37,"/"
162 X=F*32+1
163 PRINT@X-32,"    "
165 PRINT@X,"---"
170 PRINT@X+32,"  /"
175 PRINT@X+64," / "
180 PRINT@X+96,"/  "
195 PRINT@37,"-"
196 PRINT@360,E$;N
197 PRINT@279,"  "
198 IFF=5THEN335
200 NEXT:IFF=5THEN162
335 FORY=258TO418STEP32
340 PRINT@Y,"0"
342 PRINT@Y," "
345 NEXT
348 PRINT@418,"0"
350 PRINT@360,D$;A;CHR$(32)
355 FORT=1TO5000:NEXT
360 CLS
365 RUN
```

WORD PROCESSOR.

To the beginner this sounds a complicated piece of machinery but it is not, so I will give a short description of it.

With a word processor, you can write letters, assignments, recipes, notes, stories for magazines and so on.

This book and my LE'VZ newsletter are written using the Dick Smith Electronics tape Word Processor. It is really quite an advanced unit, written in Machine Language so is quite fast in use. You type as you would on a type-writer but if a mistake is typed, you just correct it and continue. Characters, lines, paragraphs or whole pages of text can be inserted, deleted, moved or copied from anywhere to anywhere within seconds. The same facilities apply to a printer or tape.

The format to a printer can vary also. Left margin, width of page, right justification or wragged, double spacing and so on.

A word can be searched and replaced by another one. IE. the word "Holden" could be replaced by "Ford" in all or some of the text. And so on, too much to describe fully here. Ask your friendly D.S.E. staff to demonstrate it to you.

EXTENDED BASIC.

There are many more BASIC commands/statements that can be implimented by the use of Steve Olney's Extended Basic tape unit. The commands and routines exist in the ROM/S but for various reasons are not directly accessable to the user. The Extended Basic unit checks for the size of the VZ's memory and allows you to use about twenty five more commands. TUE $15.00.

The Tandy book which would be hard to obtain now called "LEARNING TRS 80 BASIC FOR MODELS 1, 11/16 AND 3 BY DAVID A.LIEN" is about the best text book to teach you Basic programming. It contains information on the Extended Basic commands/statements.

HI-RES GRAPHICS GEOMETRIC PLOTTING.

(A PLEA FOR MORE READABLE BASIC PROGRAMS)

The following program is a simple line plotting routine using e hi-res graphics screen. It was written to try and demonstrate how ogramming skills can be improved by following a few simple guidelines.

Unfortunately published programs in magazines are generally or examples of how to develope good programming style. A number of may have taken the trouble to to enter a listing from a magazine - t upon running the program have found that all is not well with the de! A long, tedious and frustrating session-of understanding the orly constructed code, determining all the twists and turns of the ogical spaghetti' and debugging-commences. A usual remedy is to -write the program from scratch. Not a very efficient process!

The program below is

1. Clearly coded and set out - an enormous help in UNDERSTANDING.
2. The program is STRUCTURED - a good algorithm is selected and the program 'flows' through initialization to input, procedure and output sections.
3. Loops are indented for ease of identification and nesting.
4. Naming of variables is meaningful to assist maintenance and debugging.
5. Integer storage is used where appropriate.
6. No abbreviated forms of BASIC statements are used.
7. Remarks are liberally sprinkled throughout to aid clarity.
8. Error capture and range checking on all input variables prevents program from crashing.

Clear readable code is more important than the execution speed storage requirements of the program - interpreted BASIC runs like a red snail in any case!

These guidelines should lead to code that is easier to read, derstand and debug. This leads to easier maintenance, updating or pansion of your routines as your programming skills develope.

```
10 REM ******************************
20 REM PLOT A SET OF UP TO 20 LINES      Introduction to program,
30 REM USING THE HI-RES SCREEN.          version and author.
40 REM R.B.KITCH 22/10/85
50 REM ******************************
100 REM DIM STORAGE VECTORS X% & Y%
110 DIM X%(20), Y%(20)                   Vectors to hold end coordinates
120 REM ***ACCEPT INPUT AND CHECK*****   of LN% lines - LN%+1 points.
130 PRINT"HOW MANY LINES - MAX 20":
    INPUT LN%
140 IF LN%<1 OR LN%>20 THEN GO TO 130    Test input is not over-ranged.
150 FOR I% = 0 TO LN%                    Loop for LN%+1 X-Y points.
160     PRINT"ENTER X-VAL  0-127":
        INPUT X%(I%)
170     IF X%(I%)<0 OR X%(I%)>127
        THEN GO TO 160                   Check value not off screen.
180     PRINT"ENTER Y-VAL  0-63":
        INPUT Y%(I%)
190     IF Y%(I%)<0 OR Y%(I%)>63
        THEN GO TO 180                   Check value not off screen.
200 NEXT I%                              End of input loop.
300 REM***SET UP SCREEN AND MAIN LOOP*
310 MODE(1)                              Switch screen to hi-res.
320 FOR I% = 0 TO LN%-1                  Initialize main loop for lines.
330     X1%=X%(I%):X2%=X%(I%+1)          Assign end points of line to
340     Y1%=Y%(I%):Y2%=Y%(I%+1)          temporary variables.
```

```
350 REM ***ARE POINTS THE SAME?*******
360     IF X1%<>X2% OR Y1%<>Y2% THEN   End points the same so PLOT
        GO TO 410                      point.
370     SET(X1%,Y1%):GO TO 710         Pick up another line.
400 REM ***CALC X AND Y DIFFERENCE****
410     DX%=X2%-X1%:DY%=Y2%-Y1%        Change in X and Y direction
420 REM ***SEE WHICH IS LARGER********
430     IF ABS(DX%)>ABS(DY%)THEN       Branch according to which
        GO TO 610                      difference is larger.
500 REM ***INCREMENT IY***************  Increment along Y-axis.
510     YS%=SGN(DY%):DG=DX%/DY%        Sign of STEP and GRADIENT.
520     XO=X1%+0.5                     X-axis OFFSET.
530     FOR IY% = Y1% TO Y2% STEP YS%  Initialize loop.
540         TP=(IY%-Y1%)*DG+XO         Temporary real X-value.
550         IX%=INT(TP)                Integer X-value.
560         SET(IX%,IY%)               PLOT point.
570     NEXT IY%                       END loop.
580 GO TO 710                          Pick up another line.
600 REM***INCREMENT IX***************  Increment along X-axis.
610     XS%=SGN(DX%):DG=DY%/DX%        Sign of STEP and GRADIENT.
620     YO=Y1%+0.5                     Y-axis OFFSET.
630     FOR IX% = X1% TO X2% STEP XS%  Initialize loop.
640         TP=(IX%-X1%)*DG+YO         Temporary real Y-value.
650         IY%=INT(TP)                Integer Y-value.
660         SET(IX%,IY%)               PLOT point.
670     NEXT IX%                       END loop.
700 REM***END LOOP FOR LINE***********
710 NEXT I%:SOUND 0,9                  END main loop and PAUSE.
800 REM ***GO AGAIN?******************
810 PRINT" (E) TO EXIT"                Screen message or MENU.
820 PRINT" (P) TO PLOT AGAIN"
830 PRINT" (N) FOR NEW POINTS"
840 INPUT AN$                          Accept response.
850 AN$=LEFT$(AN$,1)                   Accept leftmost character
860 IF AN$="E"THEN STOP                Logical end of program.
870 IF AN$="P"THEN GO TO 310           Go back and PLOT again.
880 IF AN$="N"THEN GO TO 130           Go back for more input.
890 GO TO 810                          Wrong response.
900 END                                Physical end of program.
```

Lines 300-710 are a general purpose line plotting routine similar to the PLOT command on a MICROBEE.

WARNING !!!

WHEN UNPLUGGING ANY PIECE OF EQUIPMENT OF THE VZ, AND PLUGGING IN ANY PIECE OF EQUIPMENT INTO THE VZ, ALWAYS SWITCH THE VZ POWER OFF.

SERIOUS DAMAGE CAN RESULT IF THIS IS NOT DONE.

... BASIC by Bob Kitch

It may be a surprise to most BASIC programmers but the FUNCTION command, along with SUBROUTINES, are probably the most useful commands. They are concise and clarify coding considerably. Unfortunately only SUBS are supported on the VZ.

I have also had many queries from Users on how to use the FUNCTION statement in program conversions. Read on...

Level II BASIC supports two types of function -

1. library (or system) functions.
2. user-defined functions.

Functions can be used to manipulate numeric or string data types. The VZ supports a number of intrinsic or library functions such as SQR, ATN, RND, CHR$, LEFT$ and INT etc. The procedures for these are imbedded in the ROM, as BASIC utilities. Steve Olney's Extended BASIC "wakes up" a few more, such as DEFINT, CSNG and STRING$.

Unfortunately one of the omissions from the full Level II implementations on the VZ is that user defined functions are not supported in any way. Note that functions only return a single value to the program.

The lack of this feature often crops up when attempting to convert programs to run on the VZ - but written in other dialects of BASIC. The concise coding inherent in function statements is also a desirable feature. Fortunately a Fairly simple remedy is at hand and described below.

The function statement has two components. The first is the definition of the function, and the second is the actual implementation or call to that definition. Let's explain..... Suppose we wish to frequently compute the area of a circle given a number of values for the radius. The command line

```
10 DEF FNA(R)=  3.1416*R*R
```

should be declared early in the program, where DEF means define, FNA means function A (any letter from A to Z can be used to identify the particular function) and (R) is the dummy argument (for radius) used by the function. The right hand side of the assignment is the easily recognized formulae for calculating area of a circle.

Later in the program when various values are assigned to V (either from DATA or INPUT statements) we actually calculate the area by calling the procedure as follows

```
200 PRINT V,FNA(V)
```

The radius followed by the corresponding area will be written out.

As already stated, this neat construct does not exist in VZ BASIC. Judicious use of the SUBroutine statement can overcome this shortfall however. Although the function calls can only return a single value, the SUBroutine can return many values - but a few more assignments are required before going to the subroutine.

An example best illustrates this - let's use the previous example to show how it CAN be implemented on the VZ. ...

```
10 INPUT"ENTER RADIUS OF CIRCLE",R
20 GOSUB 1000
30 PRINT"RADIUS";R,"AREA";A
40 GO TO 10
1000 A=  3.1416*R*R
1010 RETURN
```

Not too difficult to set up is it? But the coding and program flow is not quite as clear.

Have fun ! and don't be foxed by functions when next converting BASIC programs onto the VZ.

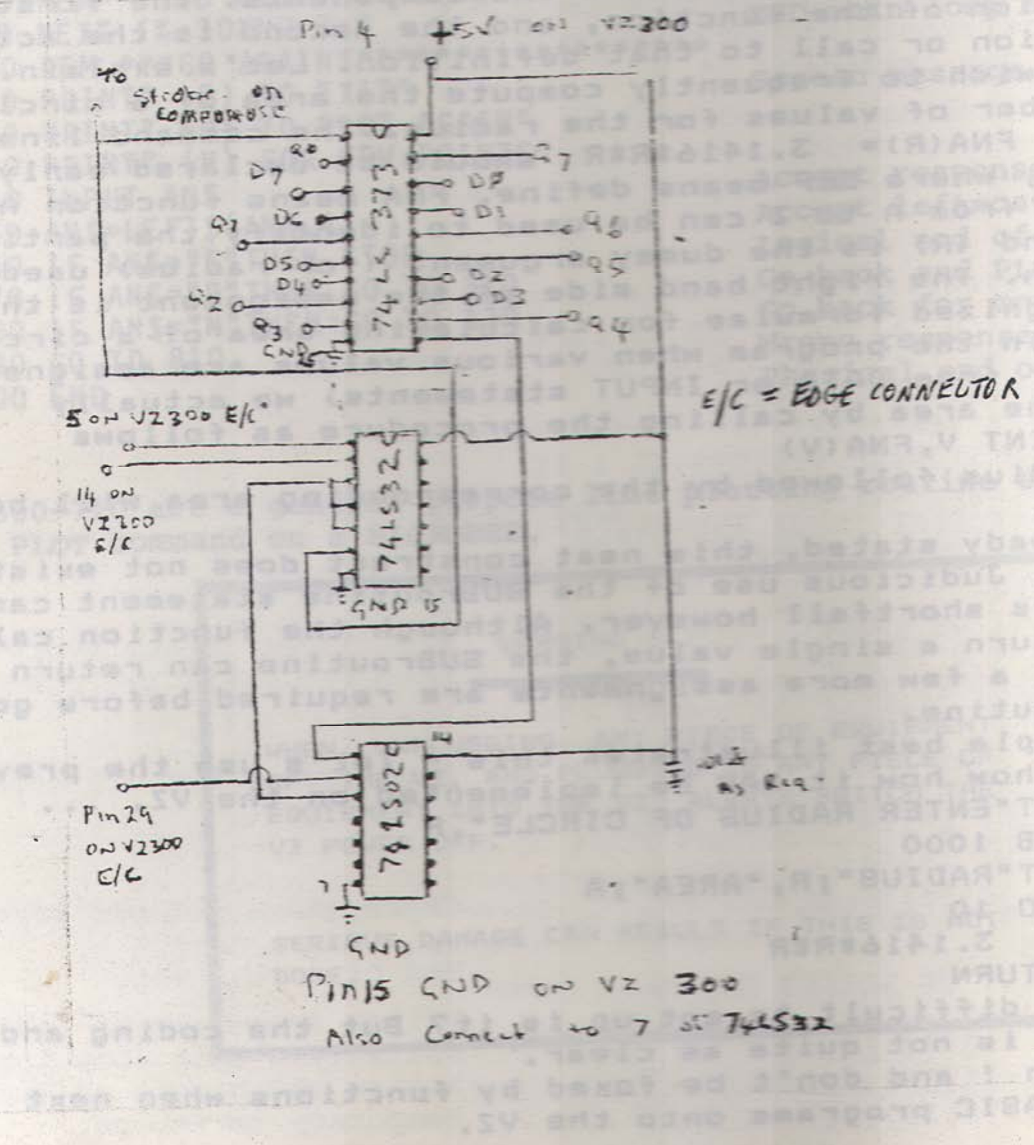
* * INTERFACE FOR COMPUMUSE SYNTHESISER * *

Some folk are having trouble when running the Compumuse unit via the Printer Interface. As the connections at the "D" plug which plugs into the printer, or in this case the Compumuse unit, are not a standard Centronics interface, modifications to either are necessary. Also there appears to be at least two different versions of the Printer Interface, which affects the OUT command. Mr. Hall designed this extra little unit which latches the OUT command signal.

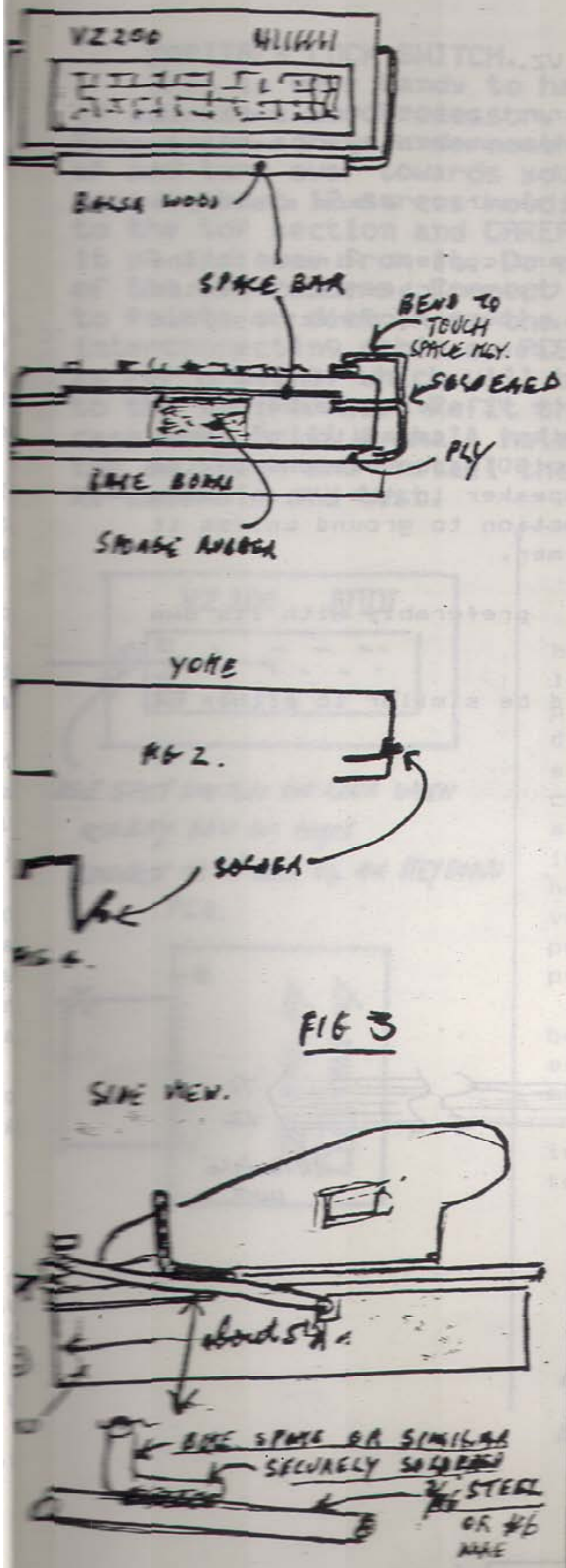
VZ300 interface addressed by :- OUT,7,XX

The edge connector contacts of the Printer/Joystick socket looking from the top of the VZ300 with keyboard in front of you as normal operation are :- Pin 1 is top row left, pin 15 is top row right, Pin 30 is bottom right. This interface will control the Compumuse as described in Electronics Australia. Change the Basic listing to address Port 7 IE. N=7 OUTN,XX. It could be used to control eight devices by fitting driver transisters to Q1 to Q7, IE. OUT7,1, OUT7,2 and so on.

WARNING!! Some plug power packs as suggested to power the Compumuse unit are only halfwave rectified and poorly filtered so hum may be present, and so distorting the sound. This may also cause the video to be shakey.



FITTING A SPACE BAR TO THE VZ200.



OK! You want to "Hack" so try this for size!

Getting tired of not finding a space bar in the right place I tried this:-

You need a baseboard 12 inch square, and a piece of masonite or ply the same size.

About 5 inches from one edge of the baseboard, cut a slot say 3/16 in. wide by 5/16 deep right across the baseboard.

Now a piece of rod 3/16 in dia. and about 25 inches long. I used a piece of #6 fencing wire. Bend it as in fig.2.

Next assemble .12" baseboard, the piece of bent wire, I'll call it a yoke, then the ply-masonite, and the V.Z. fig.3.

As the V.Z is not fastened down all measures are approx.

Next another piece of wire, I used a piece of a bike spoke, is cut and bent something like fig.4. It has a tail bent to lie along the yoke and then rise above the keyboard by about 1/4 in. and reach over to the space key and bend down to just clear the space key, with the yoke 3/8 in. off the baseboard.

Then solder the tail of this piece to the yoke. Now bend this piece so the point just clears space. A piece of sponge rubber under yoke holds it thus and acts as spring. when bending this piece use 2 pair of pliers so the strain is not taken on the soldered joint.

Now a piece of light wood (i used Balsa wood) the width of the computer and about 3/8" by 1/4". This fastens on the yoke as the thumb pad. I used hot melt glue to glue it to the steel yoke.

If you want a clear board to use the arrow keys in games, just fold it over the top and let it rest on the back of the computer case.

END

QUICK AND EASY INPUT TO THE VZ.

If you would like to be able to connect one to five switches that would signal the VZ to print or save something to be later used then this is the simplest way of all.

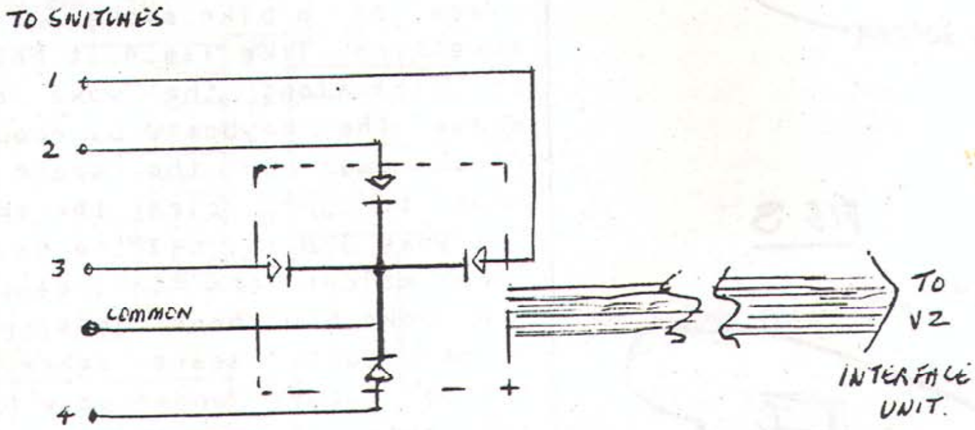
The switches could be part of a security alarm system, a doorchime system, etc.

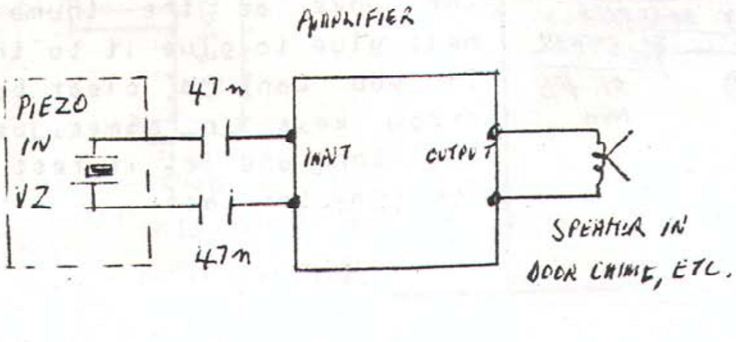
Open one of the Joystick units and connect wire/s to the outside connection/s. The common of all the switches is connected to the centre contact. In other words, you are connecting your switches in parallel to the Joystick switches.

If you want to feed the sound from the VZ piezo speaker to an amplifier for an alarm or doorchime system, a capacitor of about 47n (.047) 64 volts must be in series to BOTH connections to the amplifier. This is because the piezo speaker in the VZ is above ground. Most amplifiers have one connection to ground unless it has a balanced ungrounded input transformer.

Any amplifier would be suitable, preferably with its own power supply.

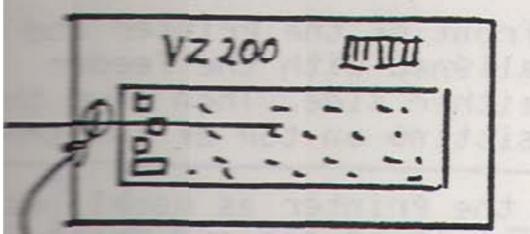
Programming the switch input could be similar to either of the listings elsewhere in this book.





CAPITALS LOCK SWITCH.

This is very handy to have if you use a wordProcessor. Remove the screws underneath, lift toP section uP and turn over towards you onto bench, remove about 12 screws'holding the keyPad to the toP section and CAREFULLY hinge it uP and away from it. Do not loose locations of the key rubbers. Connect the switch to Points on diagram on the keyPad interconnecting cable on PCB. edge 1 and 14 as Per drawing, which will be in Parrallel to the <SHIFT> key. Refit the keyPad to case toP. Drill a small hole in the case toP as shown and install the switch. Re-assemble and test.



USE SPDT SWITCH IN LINE WITH QWERTY ROW OF KEYS
CONNECT TO 1 AND 14 ON KEYBOARD PCB.

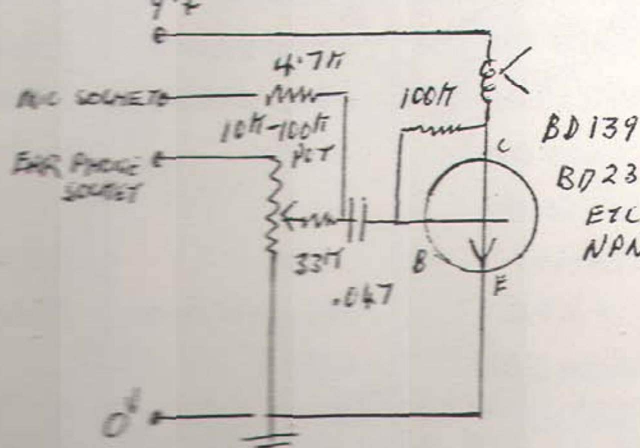


DATA RECORDER SOUND MONITOR.

It is very handy to be able to hear the computer sounds when loading and saving data and programmes. A small hole, 10 MM diameter somewhere on the top surface of the DTR between the tape counter and the rear edge will allow the sound to be heard. The 100K pot can be mounted near the hole, although control of the volume is not eccsential, so a tab pot can be mounted inside and pre-adjusted.

The sound emitting device can be a dynamic microphone insert, an earphone insert or similar unit of at least 200 Ohms impedance.

A small tag strip mounted inside connects the components together.



** A SINGLE SHEET FEEDER FOR YOUR GP 100 PRINTER **

This very simple device, which I threw together one afternoon bits & pieces I found in the shed, will enable you to print on s sheet paper and is especially useful for letterhead paper as the 2" or so of the paper cannot be printed on.

The device is basically a pair of soft rubber rollers mounted an arm. Tension is applied to the arm (in this case with a rubber band) so the paper is pinched between the guide rollers on the pr sprocket shaft and the rubber rollers. The paper is thus pulled p the the print head as the sprocket shaft turns.

Construction should be pretty straight forward using the draw as a guide. For the rollers and spacers I used plastic "COATS" co reels, with "Bradford" rubber pipe insulation on the reel for the rollers. Of course, anything that would have sufficient "grip" on paper should suffice.

The knobs are a couple of old radio knobs I found in my junk but initially I used a couple of clothes pegs to stop everything falling off the ends.

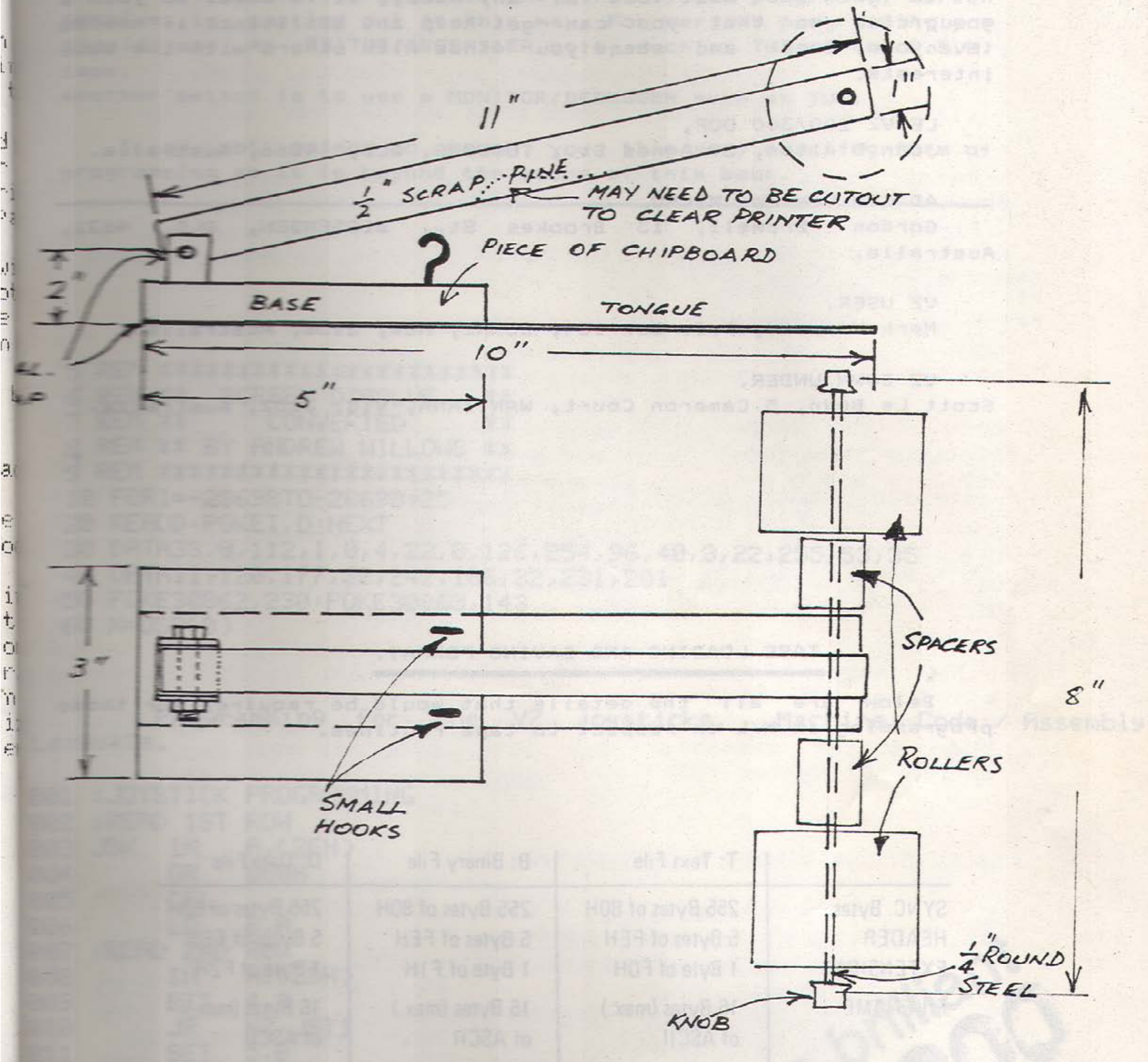
To use, slide the tongue under the front of the printer and the sprocket shaft rollers so they are aligned with the feeder rollers, pushing the feed sprockets to either side. Then move the feeder in or out until the rollers are sitting on top of the spro shaft rollers.

Feed your paper in from the back of the printer as usual, us the guide lines on the paper chute to keep it square and bring i between the rollers. Lastly, place a fairly solid rubber band fr hook, over the arm to the other hook to pull the rollers together

I have been using foolscap size paper for my letters and the trimming the top off with a razor blade to bring it back to A4 s but that's only because I obtained a few reams of white bond pap that size. If you're in the same boat, then this gadget might be what you're after.

Happy Printing,

BOB SMALL


11"
1"
1/2" SCRAP PINE
MAY NEED TO BE CUTOUT
TO CLEAR PRINTER
PIECE OF CHIPBOARD
2"
BASE
TONGUE
10"
5"
8"
3"
SPACERS
ROLLERS
SMALL
HOOKS
1/4" ROUND
STEEL
KNOB

USER GROUPS OR CLUBS.

To get the most out of any hobby, it is usual to join a group/club so that you can get help and assistance if needed (everyone does) and share your finds with others with the same interests.

LE'VZ 200/300 OOF,
John D'Alton, 39 Agnes St., TOOWONG, QLD, 4066, Australia.

AD LIB Vee Zed MICRO,
Gordon Browell, 13 Brookes St., BIGGENDEN, QLD, 4621, Australia.

VZ USER.
Mark Harwood, P.O. Box 154, DURAL, NSW, 2158, Australia.

VZ DOWN UNDER.
Scott Le Brun, 5 Cameron Court, WANTIRNA, VIC, 3152, Australia.

TAPE LOADING AND SAVING FORMAT.

Below are all the details that would be required for those programming in M/L in respect to tape routines.

	T: Text File	B: Binary File	D: Data File
SYNC. Bytes	255 Bytes of 80H	255 Bytes of 80H	255 Bytes of 80H
HEADER	5 Bytes of FEH	5 Bytes of FEH	5 Bytes of FEH
EXTENSION	1 Byte of F0H	1 Byte of F1H	1 Byte of F2H
FILENAME	16 Bytes (max.) of ASCII	16 Bytes (max.) of ASCII	16 Bytes (max.) of ASCII
GAP	3 ms Blank	3 ms Blank	3 ms Blank
START ADDRESS	2 Bytes of binary	2 Bytes of binary	----
END ADDRESS	2 Bytes of binary	2 Bytes of binary	----
Program Content	xx Bytes	xx Bytes	----
Data Content	----	----	xx Bytes
Checksum	2 Bytes	2 Bytes	2 Bytes
End of File Marker (EOF)	20 Bytes of Zeroes (00H)	20 Bytes of Zeroes	----
Terminator	----	----	1 Byte of 00H

MACHINE and ASSEMBLY PROGRAMMING.

The easiest way to start this method of programming is to POKE the values in DECimal values equivelant to HEX values into memory by a Basic programme. The first little programme, Screen Desolve is carried out in this way. For serious programming you should use an EDITOR/ASSEMBLER unit, such as TU2 or the D.S.E. tape.
Another method is to use a MONITOR/DEBUGGER such as TU9.

I am not going to teach you this very exacting form of programming as it is beyond the scope of this book.

```
5 REM **********************
6 REM **  SCREEN DISOLVE   **
7 REM **    CONVERTED      **
8 REM ** BY ANDREW WILLOWS **
9 REM **********************
10 FORI=-28698TO-28698+25
20 READD:POKEI,D:NEXT
30 DATA33,0,112,1,0,4,22,0,126,254,96,40,3,22,255,53,35
40 DATA11,120,177,32,242,186,32,231,201
50 POKE30862,230:POKE30863,143
60 X=USR(0)
```

Programming for the VZ joysticks. Machine Code / Assembly Language.

```
;JOYSTICK PROGRAMMING
;READ 1ST ROW
JSK   IN   A,(2EH)
      OR   0E0H
      CPL
      LD   B,A
;READ 2ND ROW
      IN   A,(2DH)
      BIT  4,A
      JR   NZ,JST1
      SET  5,B
;READ 3RD ROW
JST1  IN   A,(2BH)
      OR   0E0H
      CPL
      LD   C,A
;READ 4TH ROW
      IN   A,(27H)
      BIT  4,A
      RET  NZ
      SET  5,C
      RET
```

This routine reads the status of both joysticks and returns with the results in the B and C registers. The appropriate bit is set to logic 1 if that joystick is enabled, except that the "fire" switches are transferred to bit 5.

TWO M/L "PATCHES" TO ALLOW A PRINTER TO WORK WITH THE D.S.E. EDITOR/ASSEMBLER UNIT.

There appears to be more than one version of the D.S.E. unit, as my GP100 operates O.K. The first patch was sent by Jamie Perry of the D.S.E. Hot LINE.

The second from DR.P Thursby.

Below is a patch to enable your editor assembler to list source code. As stated in the manual using option C.
First enter Insert mode by entering 'I'. Then set code origi entering 'O'. Now type in the below program, pressing RETURN at end of each line.

```
     LD   BC,0CH       ;Size of transfer is 12 bytes.
     LD   HL,LOOP      ;Point to new printer routine
     LD   DE,8F54H     ;Point to editor assembler print out
     LDIR              ;Transfer routine to editor assembler
     JP   7B00H        ;Return control to editor assembler
LOOP IN A,(00H         ;Load printer status
     BIT 0,A           ;Check ready bit
     JR NZ,LOOP        ;Repeat LOOP if not ready
     LD A,C            ;Load Accumalator with print data
     OUT (0EH),A       ;Output data to printer port
     OUT (0DH),A       ;Another port for an early interface
     RET               ;Get next character
```

Now assemble the program by entering 'A'. Now RUN the progr entering 'R' then press 'Y' to verify you wish to execut program. Finish up by deleteing the program by entering Your editor assembler may list programs now, just by sele option 'C'.(enter 'SC').

```
     ;*** TEST PROGRAM 1 ***
     ;
     ;  P.THURSBY 12/85
     ;TO USE CHAR OUT ROUTINE
     ;ON VZ300 COMPUTER.
SOUT EQU  33AH
CLR  EQU  1C9H
EDIT EQU  7B00H'
     ;
     ;SAVE ALL REGISTERS
STRT PUSH AF
     PUSH DE
     PUSH HL
     PUSH BC
     CALL CLR
     POP  BC
     POP  HL
     POP  DE
     POP  AF
     ;NOW FOR SOUT ROUTINE
     PUSH BC
     LD   B,255
LOOP LD   A,24H
     CALL SOUT
     DJNZ LOOP
     POP  BC
     JP   EDIT
     ;JUMP TO EDITOR/AS
     ;ASSEMBLE AT "O <
```

------A THOUSAND VZ SCREENS-----

To demonstate how quickly Z80 Assembler can fill the screen the following program was written. It also demonstates how different background colours, colour sets and modes are implemented on the VZ. To really make the program move along change line 62 to D=1. Have fun working out the program.

```
5                                  '*** VZ-300 INSTANT COLOR ***
6                                  '***    AFC AUGUST '85    ***
7                                  '***  R.B.KITCH 18.5.86   ***
8 '
9 '***LOAD MACHINE CODE.***
10 FOR I=-28687 TO -28674
20      READ A:POKE I,A
30 NEXT I
39 '
40 DATA 33,0,112        :'LD HL,7000H    (#28672D START VIDEO RAM)
41 DATA 17,1,112        :'LD DE,7001H    (#28673D NEXT)
42 DATA 1,255,7         :'LD BC,07FFH    (#2047D  SIZE OF VIDEO RAM)
43 DATA 54,85           :'LD (HL),55H    (#85D YELLOW OR CHAR "U")
44 DATA 237,176         :'LDIR           (BLOCK LOAD COMMAND)
45 DATA 201             :'RET
46 '
49 '***INITIALIZE USR() TO ADDRESS 8FF1H OR #-28687D***
50 POKE 30862,241:POKE 30863,143
60 '***INITIALIZE DELAYS.***
61 T=0  :'***TONE  0 IS REST.
62 D=9  :'***DURATION  9 IS LONG.
63 '***SET UP DEMO LOOP.***
64 FOR I=0 TO 255
65      POKE -28677,I      :'***OVERWRITE WITH NEW CHARACTER.***
66      '***SCREEN MESSAGE.***
67      MODE(0):PRINT@234," CHAR = ";I:SOUND T,D
68      X=USR(0)           :'***FILL 2K VIDEO RAM WITH CHAR.***
69      '***LO-RES GREEN BACKGROUND.***
70      COLOR,0:SOUND T,D
79      '***LO-RES ORANGE BACKGROUND.***
80      COLOR,1:SOUND T,D
89      '***HI-RES COLOR SET 1.***
90      MODE(1):X=USR(0):'***FILL AGAIN AFTER RESET.***
91      COLOR,0:SOUND T,D
99      '***HI-RES COLOR SET 2.***
100     COLOR,1:SOUND T,D
110 NEXT I
120 STOP:END
```

This program looks for a specified byte. Once it is found th program backspaces to the previous byte and then prints th contents of the address being pointed to, in HEX to the printer. The search covers the entire ROM and the DOS region. In this cas I was searching the contents for the actual Communication addresses in the range from 7AØØH to 7AFFH.

```
        CALL 3AE2H      ;If no printer change to CALL Ø1C9H
        LD   BC,6ØØØH
        LD   HL,ØØØØH
   RETN LD   A,(HL)
        CP   7AH
        JR   NZ,NEXT
        PUSH BC
        PUSH HL
        DEC  HL
        LD   B,(HL)     ;Save the low byte contents in B
        INC  HL         ;Move to the next byte
        LD   A,(HL)     ;Load A with the high byte contents
        CALL HEX
        LD   A,B        ;Load A with the low byte contents
        CALL HEX
        LD   C,32       ;If no printer change to LD A,32
        CALL Ø58DH      ;If no printer change to CALL Ø33AH
        POP  HL
        POP  BC
   NEXT INC  HL
        DEC  BC
        LD   A,B
        OR   C
        JR   NZ,RETN
        CALL 3AE2H      ;If no printer then omit this line
        JP   31488      ;If assembling change to JP 1A19H
   HEX  PUSH AF
        RRCA
        RRCA
        RRCA
        RRCA
        CALL HEX2
        POP  AF
   HEX2 AND  ØFH
        ADD  A,3ØH
        CP   3AH
        JR   C,DISP
        ADD  A,7
   DISP PUSH HL
        LD   C,A
        CALL Ø58DH      ;If no printer change to CALL Ø33AH
        POP  HL
        RET
```

This program searches for a pair of bytes, that is, an address. Once found the location containing the low byte of the pair is printed in HEX to the printer. The search covers the entire ROM and the DOS region. In this case I was searching for any reference to 7AE9H, the start of Basic pointer.

```
     CALL 3AE2H
     LD   BC,6000H
     LD   HL,0000H
RETN LD   A,(HL)     ;Load A with the contents of HL
     CP   0E9H       ;Check to see if it is equal to E9H
     JR   NZ,NEXT    ;If not go on to the next byte
     INC  HL         ;If yes move on one place
     LD   A,(HL)     ;Load A with contents of new place
     CP   7AH        ;Check to see if contents equal to 7AH
     JR   NZ,NEXT    ;If not go on to next byte
     PUSH BC
     PUSH HL
     DEC  HL
     LD   B,L        ;Save the low byte contents in B
     LD   A,H        ;Load A with the high byte
     CALL HEX
     LD   A,B        ;Load A with the low byte contents
     CALL HEX
     LD   C,32
     CALL 058DH
     POP  HL
     POP  BC
NEXT INC  HL
     DEC  BC
     LD   A,B
     OR   C
     JR   NZ,RETN
     CALL 3AE2H
     JP   31488
HEX  PUSH AF
     RRCA
     RRCA
     RRCA
     RRCA
     CALL HEX2
     POP  AF
HEX2 AND  0FH
     ADD  A,30H
     CP   3AH
     JR   C,DISP
     ADD  A,7
DISP PUSH HL
     LD   C,A
     CALL 058DH
     POP  HL
     RET
```

Enhancing VZ Basic by Larry Taylor

The Commodore 64 has advanced hardware supported by an inadequate Basic language, resulting in a number of enhanced Basics being available. Something similar could be produced for the VZ. It must be noted, however, that all such Basics share a common disadvantage. Any program which makes use of them requires the language be loaded before it will function properly.

Because Basic is an interpreted language additional commands can be inserted, if they can be intercepted and executed before reaching the VZ's own interpreter. This is precisely what happens when a disk operating system (DOS) is added. New commands enabling disk operations to be performed, supplement the existing Basic. However, all programs using those extra commands require the DOS to be present before execution or they will not be interpreted correctly.

When a Basic program is RUN, control passes to a machine language ROM routine, the Execution Driver at 1D5AH, which scans each line of the Basic program as it comes to it and begins to translate it. Part of the translation process involves looking for tokens. These are values in the range 128-250 (80H-FAH) that take the place of Basic reserved words e.g. CLS = 132 (84H). Once the word has been identified and checked for correct syntax, control is passed to the corresponing ROM routine before returning to continue the translation. This is similar to one person issuing instructions to another through an interpreter, who first has to translate them before the receiver can act, and is the reason for Basic's slow execution. Most languages get around this problem by having the program translated or compiled before execution.

Tandy's Colour Computer has an enchanced CLS command which enables the user to clear the screen to any one of nine background colours. The syntax is CLSn, where n may be a number in the range 0-8. To illustrate how enhancements can be accomplished, this command will be added to the VZ's repertoire.

On power up the address of the routine which examines each byte in a line of Basic, is stored at 7804H. Because this address is in RAM it can be easily changed. This was done so that at a later stage the DOS could be included. However, it also means that, just as readily, an enhanced form of Basic may be added. The trick is to ensure that, as far as the VZ's interpreter is concerned, nothing unusual has happened. The accompanying assembly language listing shows how this can be accomplished.

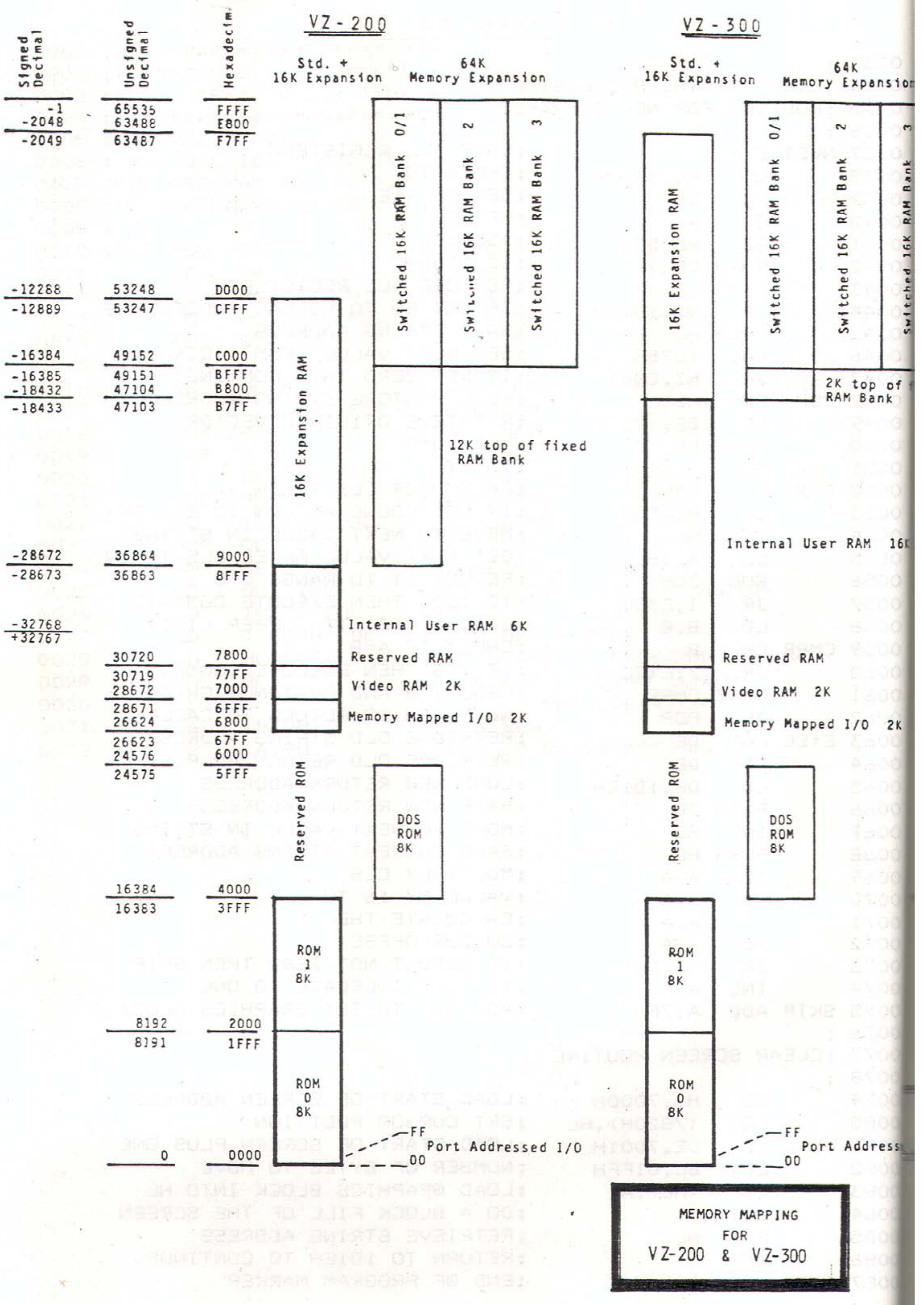
Having adjusted the top of memory pointer, the address at 7804H is stored and replaced by our own. The program then locates the new routine at the top of memory. Now each time a byte is to be examined during execution it must first pass through our checkpoint. Once the origin of the call is established, the routine looks for the CLS token, 132 (84H). Only when it has been located does the routine proceed to examine the next byte. This is checked to see if it lies in the range 0-9. Once it has passed this test, the clear screen routine is implemented after first calculating the appropriate value with which to fill the screen. You will notice that not only is it necessary to check for the new command, but also to provide the routine which implements it. In this case a simple block load to the screen has been used. Control is then returned to the ROM processing routine, which prepares to examine the byte following our new command. So, as far as the VZ knows, everything is continuing normally. Tricky isn't it?

I have already successfully used this approach to produce a VZ Printer Patch, which enables all the normal printer functions for owners of EPSON or EPSON compatible printers. The COPY command is intercepted by the patch and as a result its function has been enhanced to allow a proper dump of both the LO-RES and HI-RES screens. One further enhancement that could be explored would be an extension of Basic's SOUND command. The possibilities are limited only by imagination and memory.

```
;##########################
;# ENHANCED CLS COMMAND #
;# BY LARRY TAYLOR 1986 #
;##########################
; ORIGIN = 7B00H
;THIS SECTION RELOCATES
;THE PROGRAM TO THE TOP
;OF AVAILABLE MEMORY.
;
VCTR EQU   7A28H         ;SET VCTR AS 7A28H
     LD    SP,7700H      ;LOAD STACK POINTER
     LD    HL,(78B1H)    ;GET THE TOP OF MEMORY
     LD    BC,ENDP-NVCT  ;GET LENGTH OF PROGRAM
     PUSH  BC            ;SAVE PROGRAM LENGTH
     XOR   A             ;RESET ALL FLAGS
     SBC   HL,BC         ;TAKE LENGTH FROM TOP OF MEMORY
     LD    (78B1H),HL    ;LOAD NEW TOP OF MEMORY
     PUSH  HL            ;SAVE NEW TOP OF MEMORY
     XOR   A             ;RESET ALL FLAGS
     LD    BC,33H        ;RESERVE 50 BYTES STRING SPACE
     SBC   HL,BC         ;TAKE SPACE FROM TOP OF MEMORY
     LD    (78A0H),HL    ;LOAD START OF STRING SPACE
     POP   DE            ;RETRIEVE TOP OF MEMORY
     INC   DE            ;INCREASE BY ONE
     LD    HL,(7804H)    ;GET CURRENT RST10H VECTOR
     LD    (VCTR),HL     ;STORE IT IN 7A28H
     LD    (7804H),DE    ;LOAD NEW VECTOR
     LD    HL,NVCT       ;GET START OF PROGRAM TO MOVE
     POP   BC            ;RETRIEVE PROGRAM LENGTH
     LDIR                ;MOVE TO NEW LOCATION
     CALL  1B4DH         ;DO A NEW
     JP    1A19H         ;JUMP TO READY MESSAGE
```

FROM PAGE 42.

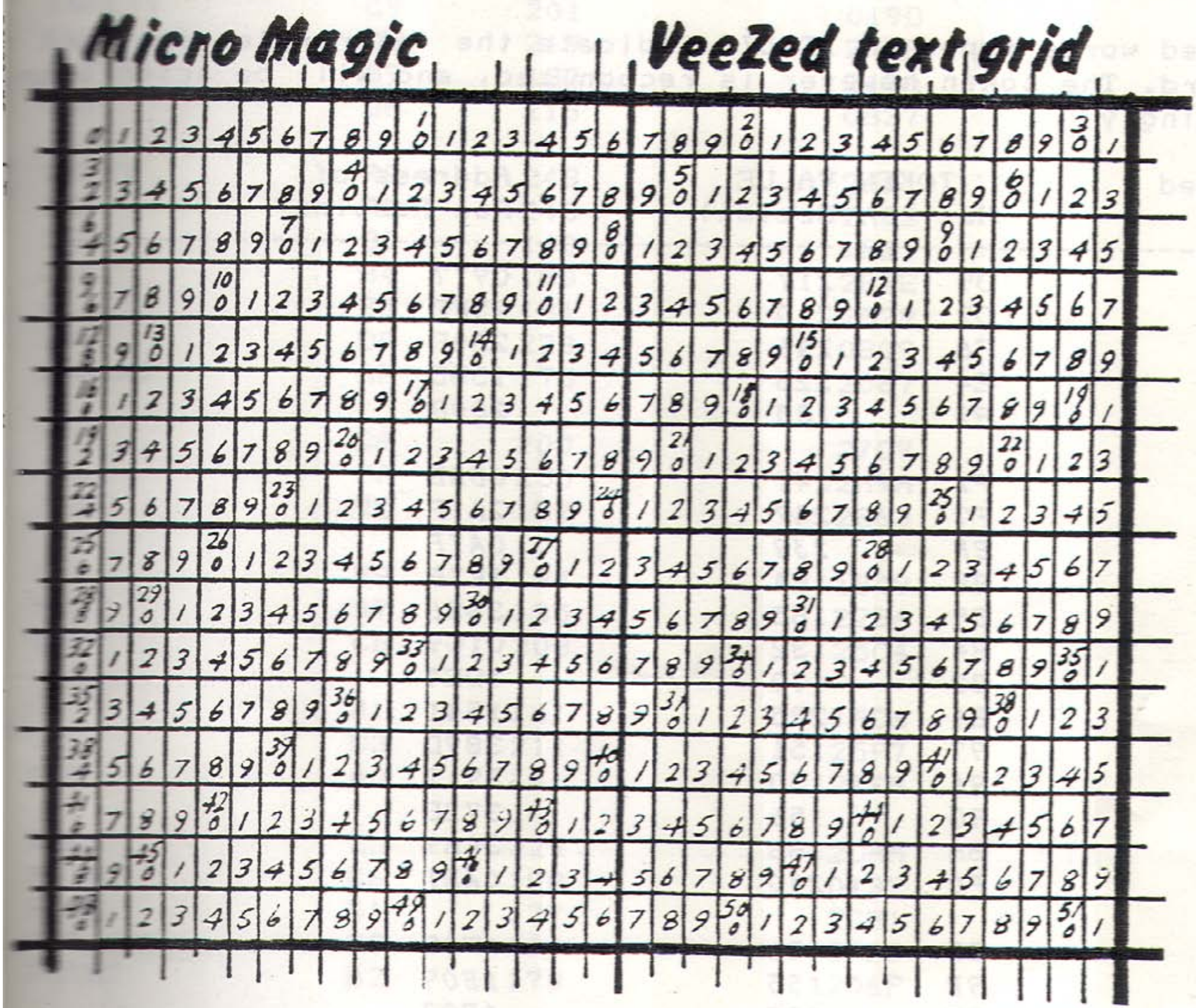
```
;
;START OF THE PROCESSING
;ROUTINE FOR NEW COMMAND.
;
NVCT EXX                ;SAVE ALL REGISTERS
     LD    HL,1D5BH     ;CHECK TO
     POP   DE           ;SEE IF THE
     OR    A            ;RETURN
     SBC   HL,DE        ;ADDRESS
     PUSH  DE           ;IS 1D5BH
     EXX                ;RESTORE ALL REGISTERS
     JP    NZ,1D78H     ;IF NOT GO TO NORMAL PROCESSING
     PUSH  HL           ;SAVE STRING ADDRESS
     CALL  1D78H        ;GET NEXT VALUE FROM STRING
     JR    NZ,CONT      ;IF NOT ZERO THEN CONTINUE
POP  POP   HL           ;ELSE RESTORE STRING ADDRESS
     LD    DE,(VCTR)    ;RETRIEVE ORIGINAL VECTOR
     PUSH  DE           ;AND JUMP
     RET                ;TO IT
CONT CP    84H          ;CHECK FOR CLS TOKEN
     JR    NZ,POP       ;IF NOT FOUND RETURN TO CALLER
     INC   HL           ;MOVE TO NEXT VALUE IN STRING
     LD    A,(HL)       ;GET NEXT VALUE AFTER CLS TOKEN
     SUB   30H          ;REDUCE IT TO RANGE 0-8
     JR    Z,EXEC       ;IF ZERO THEN EXECUTE COMMAND
     LD    B,8          ;LOAD B REG WITH UPPER LIMIT
CMPR CP    B            ;CHECK IF A=B
     JR    Z,EXEC       ;IF YES THEN EXECUTE COMMAND
     DJNZ  CMPR         ;REDUCE B AND CONTINUE CHECK
     JR    POP          ;NO MATCH SO RETURN TO CALLER
EXEC POP   DE           ;RETRIEVE OLD STRING ADDRESS
     POP   DE           ;RETRIEVE OLD RETURN ADDRESS
     LD    DE,1D1EH     ;LOAD NEW RETURN ADDRESS
     PUSH  DE           ;SAVE NEW RETURN ADDRESS
     INC   HL           ;MOVE TO NEXT VALUE IN STRING
     PUSH  HL           ;SAVE CURRENT STRING ADDRESS
     ADD   A,A          ;MULTIPLY CLS
     ADD   A,A          ;VALUE BY 16 TO
     ADD   A,A          ;CALCULATE THE
     ADD   A,A          ;COLOUR OFFSET
     JR    NZ,SKIP      ;IF RESULT NOT ZERO THEN SKIP
     INC   A            ;IF ZERO INCREASE TO ONE
SKIP ADD   A,7FH        ;ADD 127 TO GET GRAPHICS BLOCK
;
;CLEAR SCREEN ROUTINE
;
     LD    HL,7000H     ;LOAD START OF SCREEN ADDRESS
     LD    (7820H),HL   ;SET CURSOR POSITION
     LD    DE,7001H     ;LOAD START OF SCREEN PLUS ONE
     LD    BC,01FFH     ;NUMBER OF BYTES TO MOVE
     LD    (HL),A       ;LOAD GRAPHICS BLOCK INTO HL
     LDIR               ;DO A BLOCK FILL OF THE SCREEN
     POP   HL           ;RETRIEVE STRING ADDRESS
     RET                ;RETURN TO 1D1EH TO CONTINUE
ENDP DEFB  0            ;END OF PROGRAM MARKER
```



R. B. KITCH March 1

VIDEO DISPLAY WORKSHEET. MODE 0.

Make up a similar one about twice the size, marking every second square with it's number position and cover it with plastic. It can then be used when setting out LOW RES graphics or text by writing on it with a pen that can be rubbed clean with a cloth when finished.



* * * SOFTWARE * * *

We sell a large range of exclusive tape and Disc software. [illegible]ease send a large S.A.S.E. for a VLISTZ.

These are just a few that could be usefull in conjunction [illegible]h this book.

[illegible] EDITOR/ASSEMBLER $ 20.00.

[illegible] EXTENDED BASIC $ 15.00.

[illegible] MONITOR/DEBUGGER $ 25.00.

[illegible] EXTENDED BASIC $ 12.50.

This is part of the VZ communications area. It is invaluable for those who are programming in M/L.

VZ 200 / 300 COMMUNICATION AREA - RESERVED RANDOM ACCESS MEMORY

RESERVED WORD LIST

Reserved words typed in *ITALIC* indicate the interpreter does no the word. The token however is recognized, and will be acted up accordingly

Reserved word	TOKEN VALUE Hex	Decimal	Address of Rom Routine
ABS	D9	217	0977
AND	D2	210	25FD
ASC	F6	246	2A0F
ATN	E4	228	15BD
AUTO	*B7*	*183*	*2008*
CDBL	*F1*	*241*	*0DAB*
CHR$	F7	247	2A1F
CINT	*EF*	*239*	*0A7F*
CLEAR	B8	184	1E7A
CLOAD	B9	185	3656
CLS	84	132	01C9
CONT	B3	179	1DE4
COS	E1	225	1541
COLOR	97	151	389D
COPY	96	150	3912
CRUN	9C	156	372E
CSAVE	BA	186	34A9
CSNG	*F0*	*240*	*0AB1*
DATA	88	136	1F05
DEFDBL	*9B*	*155*	*1E09*
DEFINT	*99*	*153*	*1E03*
DEFSNG	*9A*	*154*	*1E06*
DEFSTR	*No recognized token*		*1E00*
DELETE	*B6*	*182*	*2BC6*
DIM	8A	138	2608
ELSE	95	149	1F07
END	80	128	1DAE
ERL	*C2*	*192*	*24DD*
ERR	*C3*	*193*	*24CF*
ERROR	*9E*	*158*	*1FF4*
EXP	E0	224	1439
FIX	*F2*	*242*	*0B26*
FOR	81	129	1CA1
FRE	*DA*	*218*	*27D4*
GOSUB	91	145	1EB1
GOTO	8D	141	1EC2

VZ 200 / 300 COMMUNICATION AREA - RESERVED RANDOM ACCESS MEMORY

Reserved word	TOKEN VALUE Hex.	Decimal	Address of Rom Routine
IF	8F	143	2039
INKEY$	C9	201	019D
INP	DB	219	2AEF
INPUT	89	137	219A
INT	D8	216	0B37
LEFT$	F8	248	2A61
LEN	F3	243	2A03
LET	8C	140	1F21
LIST	B4	180	2B2E
LLIST	B5	181	2B29
LOG	DF	223	0809
LPRINT	AF	175	2067
MEM	*C8*	*200*	*27C9*
MID$	FA	250	2A9A
MODE	9D	157	2E63
NEW	BB	187	1B49
NEXT	87	135	22B6
NOT	CB	203	25C4
ON	*A1*	*161*	*1FC6*
OR	D3	211	25F7
OUT	A0	161	2AFB
PEEK	E5	229	2CAA
POINT	C6	198	0132
POKE	B1	177	2CB1
POS	*DC*	*220*	*27F5*
PRINT	B2	178	206F
RANDOM	*86*	*134*	*01D3*
READ	8B	139	21EF
REM	93	147	1F07
RESET	82	130	0138
RESTORE	90	144	1D91
RESUME	*9F*	*159*	*1FAF*
RETURN	92	146	1EDE
RIGHT$	F9	249	2A91
RND	DE	222	14C9
RUN	8E	142	1EA3

VZ 200 / 300 COMMUNICATION AREA - RESERVED RANDOM ACCESS MEMORY

Reserved word	TOKEN VALUE Hex.	Decimal	Address of Rom routine
SET	83	131	0135
SGN	D7	215	098A
SIN	E2	226	1547
SOUND	9E	158	2BF5
SQR	DD	221	13E7
STEP	CC	204	2B01
STOP	94	148	1DA9
STR$	F4	244	2836
STRING$	*C4*	*196*	*2A2F*
TAB	BC	188	2137
TAN	E3	227	15A8
THEN	CA	202	2039
TO	BD	189	1CA1
TROFF	*No recognized token*		*1DF8*
TRONN	*No recognized token*		*1DF7*
USING	BF	191	2CBD
USR	C1	193	27FE
VAL	F5	245	2AC5
VERIFY	98	152	3738
VARPTR	*C0*	*192*	*24EB*

If you are having any problems with any article or programme in this book don't hesitate to contact me. Also for any input, suggestions etc, please write or 'phone. Any communications in writing that you require, MUST INCLUDE A S,A,S,E. with your request.

God bless . . . . John D'Alton.